UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

WLAN Fingerprinting based Indoor Positioning in the Presence of Censored and Dropped Data

Von der Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

M.Sc. Manh Kha Hoang

Erster Gutachter: Prof. Dr.-Ing. Reinhold Häb-Umbach
Zweiter Gutachter: Prof. Dr.-Ing. Peter A. Höher

Tag der mündlichen Prüfung: 04. März 2016

Paderborn 2016

Diss. EIM-E/321

Acknowledgments

First of all, I would like to give a great thanks to Prof. Dr.-Ing. Reinhold Häb-Umbach for being my research supervisor. I would have never been able to complete my research work and my dissertation without his continuous support, understanding and patience. He has kindly motivated me for the new challenges and patiently guided me to overcome the difficulties. I feel very lucky to have met Prof. Häb-Umbach and worked with him. I would also like to thank Prof. Dr.-Ing. Peter A. Höher who have evaluated my dissertation and given me the valuable responses to improve its quality.

Moreover, I would like to thank Vietnamese government for granting the scholarship for my first three years of studying in Germany. Without it, I would have never had such a good chance to commence this PhD degree in Germany. I also owe thanks to the Department of Communications Engineering, University of Paderborn, for providing me the living expense for my last two years of my research, in particular, once again thanks to Prof. Häb-Umbach.

I really appreciate all the support from all staff members of the World University Service who have introduced me to Prof. Häb-Umbach, helped me to arrange the accomodation when I first came to Germany and continuously encouraged me during my research.

Special thanks to my colleagues and students at the Department of Communications Engineering who have always supported me to proceed my research and improve my dissertation. In particular, I wish to thank Dr.-Ing Jörg Schmalenströer for all of his support, valuable technical ideas and comments, without that it would be very difficult for me to complete my work.

Last but not least, I wish to thank my wife, Thi Hien Trang Vu, and my little daughter, Tra My Hoang, for without their encouragement, support and patience, this work would have not been completed. In addition, I would like to thank my parents who always beside, continuously support and encourage me for not only these years but also whole my life.

Contents

"In the age of automation the ability to navigate persons and devices in indoor environments has become increasingly important for a rising number of applications."

Rainer Mautz [1]

"Despite its current limitations, indoor navigation's huge potential economic and sociological capabilities are pushing forward the research, development, implementation, and sale of low-cost systems. In the near future, this will change the way we interact with our surroundings, with many advantages for the various stakeholders involved in any indoor business."

Andrea Bottino, Giovanni Malnati and Paolo Montuschi [2]

1. Introduction

Positioning and *navigation* have played important roles in many aspects of human civilization for thousands of years. Demand of this service is increasing steadily in many aspects called location-based services (LBS) or location-aware systems [3, 4] such as transportation, security, social networking, marketing, and so on. While positioning or localization is the process to determine the coordinates of the target objects, navigation is the process to estimate the route from one location to another.

At the beginning, positioning and navigation techniques were investigated for outdoor environment to support the transportation. Outdoor positioning and navigation were done by a combination of celestial measurements and land marks, which helped people to control the vessel over a very large distance. With the development of technology, satellite was invented and subsequently, satellite based navigation systems such as Global Positioning System (GPS) by United States and Global Navigation Satellite System (GLONASS) by Russia were developed. At the moment, there are two other systems being developed namely Galileo by European Union and the BeiDou Navigation Satellite System by China. Among them, GPS [5, 6] is the most popular and successful navigation system. GPS was first developed for military purpose by the U.S. Army and it was made available for civilian use in the 1990s. With GPS, position estimate in three dimentions is obtained by applying a circular lateration technique, which relies on range measurements. Nowadays, GPS is used to support the transportation of most of the vehicles such as airplanes, cars and ships. Moreover, GPS can now be used on most smart devices (smartphones and tablet computers) to support the daily activities with lower positioning accuracy than the traditional GPS devices since lower-quality GPS chipsets are used on portable devices due to price limitation. In the ideal case of line-of-sight condition, i.e., no or almost no obstacles between the device and the satellites, GPS can locate a receiver with an accuracy of a few meters. However, in an urban area, the accuracy degrades dramatically due to non-line-of-sight problems.

Beside the successful GPS, information from some other sources, e.g., Global System for Mobile Communications (GSM), are also used for the outdoor positioning and navigation purposes. GSM based positioning was developed to satisfy the Enhanced 911 (E-911) mandate from the US Federal Communications Commission which requires cellular providers to track the location of their subscribers to within $50\,\mathrm{m}$ for over 67% of the time. The GSM positioning system may either solely employ the GSM information [7, 8, 9] or combine it with other sources of information, e.g., GPS, inertial sensors and WiFi, resulting in hybrid systems [10, 11] in order to produce a more accurate position estimation.

In an indoor environment, GPS signals are normally blocked or unreliable due to the attenuation of signal through roofs or walls. As a result, the positioning accuracy is poor. Therefore, developing a reliable indoor positioning and navigation system is of a particular interest. This topic has been attracted the consideration of many researchers over the last de-

cades. Many approaches emerging from different research communities such as networking, robotics, and signal processing, have been proposed [3, 12]. However, delivering an accurate position estimation with low cost and low computational costs is still a major technical challenge.

In indoor positioning, the most popular approaches to locate mobile users are lateration based, angulation based and fingerprinting based techniques using radio signals, such as Ultra-Wideband, Zigbee, Bluetooth or WiFi. Among those possible techniques, WiFi signal based techniques are of a particular interest because of the wide deployment of Wireless Local Area Network (WLAN) in many public areas such as universities, museums or hospitals. Through this dissertation, the terms WLAN and WiFi are used interchangeably, although, stricktly speaking, WiFi refers only to devices certified by the WiFi Alliance. While some lateration based and angulation based localization techniques use the Time Difference Of Arrival (TDOA) [13, 14], Time Of Arrival (TOA) [15, 16] and Angle Of Arrival (AOA) [17] which normally require additional specialized hardware, the others use the radio signal strength information, i.e., received signal strength index (RSSI), and an estimated path-loss model [18, 19] to obtain the necessary measurements for determining user position. The signal strength based approach seems to be more suitable for WLAN positioning since WiFi RSSI information can be obtained directly from existing WLAN infrastructure by any device equipped with a WLAN network adapter. However, accurate lateration based and angulation based positioning using RF signals face many challenges due to multipath effects from reflection, refraction and scattering from obstacles, walls and movement of people [18, 12, 20].

The RSSI fingerprinting based technique, which was pioneered in [18], relaxes those limitations. With the fingerprinting based techniques, a certain amount of training data are collected at possible user locations. The position estimate is determined by comparing the online observations with the training data. If a sufficient amount of training data is available, the data represents the typical variations of RSSI values and the repertoire of pattern classification techniques can be employed to arrive at an estimate of the user position.

Within RSSI fingerprinting based positioning, there are two common approaches to estimate user location: deterministic approaches, e.g., k-nearest neighbors [18, 21, 22], and stochastic approaches [23, 24, 25] which use a probabilistic model of the training data and compute the likelihood of observing the online measurements given the position dependent probability density function (PDF), or the posterior of being at a position given the online observations, to come up with the position estimate. The stochastic approaches seem to be able to efficiently cope with the variations in observed data in the training as well as the classification procedure. There are two common categories of model of the distribution of RSSI values: parametric and non-parametric models. With parametric model, a careful study on the charateristics of indoor WiFi data to find out the appropriate model is needed, and subsequently, a parameter estimation algorithm has to be developed. On the other hand, with non-parametric model, the PDF of the RSSI data is usually estimated using histogram methods. Therefore, more training data are needed to obtain a precise PDF compared to parametric model. Moreover, the database needs to store many parameters of a non-parametric model while with parametric model, it only needs to store a few parameters. In the sense of system simplicity and positioning accuracy, fingerprinting based methods are the most promising approach with the cost of more training effort.

One remaining problem of fingerprinting based indoor positioning techniques is that the training data cannot be used efficiently to locate the devices if the devices have different

charateristics from the ones observed in the training phase [26]. To develop a reliable positioning system, one must answer the question: how to cope with different portable devices when they deliver different RSSI information according to the differences of hardware and chipset drivers? The mismatch of RSSI measurements between training and test leads to positioning errors, and, hence, probably unsatisfactory positioning performance. To handle such a mismatching problem, a method to align the training models with the properties of the testing device is needed. Although this is an essential task to make indoor position systems work in reality, not many reports have mentioned this problem.

In addition, Dead Reckoning (DR) techniques [27, 28] is another successful approach for indoor positioning and navigation approach where inertial sensor data is employed. These techniques aim to determine the user position by estimating the displacement and movement heading given a known starting position. The DR techniques are able to produce precise position estimates in a short term of operation. In the long term use, the accumulated error makes the positioning estimation unreliable. To compensate the error accumulation, a correction procedure is needed to reset the error of the estimate at regular intervals using other sources of information, i.e., GPS information or WiFi data. As a consequence, for indoor environment, hybrid positioning systems which combine WiFi information and inertial sensor data have been developed to improve the positioning accuracy [29, 30, 31, 32].

Within this dissertation which aims to develop an indoor positioning and navigation system for smartphone users, approaches employing WLAN and inertial sensor information are presented. Inertial sensors such as accelerometers, magnetometers, gyroscopes, ect. are now available on most modern smartphones. It is thus possible to develop an indoor positioning system for smartphones employing the built-in sensors of the smartphones without additional hardware.

This dissertation is organized as follows:

Chapter 1 provides a very brief introduction into positioning and navigation, particularly for the indoor environment. It briefly presents the reason why in this investigation, WLAN information and inertial sensor information is employed for indoor positioning purpose.

Chapter 2 discusses the fundamental positioning in general and gives more details for the indoor environment. The principles of positioning techniques are provided here as well as their advantages and limitations. A survey of some state-of-the-art research is reported within the discussion of each technique.

Chapter 3 outlines the objectives of this thesis.

Chapter 4 presents an Expectation-Maximization (EM) algorithm for parameter estimation of censored and dropped Gaussian data. This procedure is used in the training phase of the proposed fingerprinting based indoor positioning method. The radio map which is created in the training phase does not store the raw RSSI measurement, but instead stores the statistical parameters of the RSSI distributions of all Access Points (APs) at all training positions. The proposed EM algorithm is able to cope with the censoring and dropping problem in parameter estimation of Gaussian data. An optimal classification rule within the Maximum Likelihood (ML) framework for censored and dropped data is also derived.

Chapter 5 presents a method to adapt the training models from one device to another device within the Maximum Likelihood Linear Regression (MLLR) framework. A method for estimating a regression tree of adaptation matrices is also discussed here.

Chapter 6 describes a modified Hidden Markov Model (HMM) to employ the knowledge of possible walking paths and to fuse the information from inertial sensors and WiFi APs.

Position estimation is carried out by applying a Forward Algorithm to decode the HMM. In addition, a brief discussion about step detection and movement heading estimation is provided.

Chapter 7 describes some experimental results both on simulation data and real field data in order to demonstrate the effectiveness of the proposed methods. Comparison with some other approaches are also discussed.

Chapter 8 presents the realized indoor positioning and navigation system at the University of Paderborn.

Chapter 9 concludes the work, highlights the limitations and discusses some potential future work.

2. Fundamentals of Indoor Positioning and State of Research

In this chapter, the principles of the most popular radio based indoor positioning techniques as well as some previous studies relating to each technique are discussed. The discussion focuses on the advantages and disadvantages of each technique, highlighting the reason why this study follows the fingerprinting based direction. The challenges of WiFi fingerprinting based techniques are also discussed at the end of the chapter.

Indoor positioning is a challenging research topic which has attracted a lot of research. Over the last decades, many approaches have been proposed to address those challenges in indoor environments employing different types of signals/information and the corresponding methods, as reported in [1, 3].

There are many types of signals that have been taken into consideration for indoor positioning purposes such as radio, light, sound and inertial sensor data. Many positioning approaches and systems have been built upon radio signals, i.e., Ultra-Wideband [33, 34], Zigbee [35, 13, 36], WiFi [18, 23, 24, 25, 12, 20] or Bluetooth [37]. Among them, WiFi signal based positioning methods are of particular interest since most mobile devices now are equiped with a WiFi chipset and because of the wide deployment of the WLAN infrastructure. Light and sound signals are considered for developing high precision positioning systems [38, 39, 40, 41] which are able to achieve accuracy levels of centimeters. These signals are normally used in a much smaller deployment area in comparison with radio signals while at the same time requiring more additional specialized hardware. Inertial sensor data, i.e., acceleration data, gyroscope data and/or magnetic data, were first used mainly in the robotics community to locate and track the robot position. However, with the development of Microelectromechanical systems (MEMS) technology, many portable devices are equiped with Inertial Measurement Unit (IMU), which make these data available for indoor user tracking [27, 28].

Since the goal of the project is to develop an indoor positioning system for the smartphone users within the campus of the University of Paderborn which is a large area with many multifloor buildings, light and sound signal based approaches do not seem to be suitable because many additional hardware would be needed to build the system. The same limitations hold for systems using Ultra-Wideband, Zigbee or Bluetooth signals. As a consequence, WiFi and inertial sensor data are chosen because these data can be obtained from the existing WLAN infrastructure and the built-in sensors of the smartphones.

In the following, common approaches which have been used for indoor positioning using WiFi and inertial sensor data are presented. An overview of other indoor postioning approaches can be found in [1, 3].

2.1. Proximity

Proximity based positioning techniques are the simplest positioning methods. This technique determines the position of a mobile object based on its closeness to a reference point, i.e., base station, in physical space. The operation of this method is based on the fact that the radio transmitters have a limited range and if the receiver can detect a valid signal from a transmitter, it is supposed to be in the coverage area of that transmitter, and the position of the transmitter is considered as the estimated position of the receiver.

Moreover, signal strength is considered in addition to the information of the presence of a signal if the receiver is at a position, where it can detect valid signals from more than one transmitter because it is in the overlapping area between different transmitters. The estimated position is then the position of the transmitter which has the strongest signal strength at the receiver position. This simple decision rule must use an assumption that the distance in the received signal strength space is propotional to the physical distance between the transmitter and the receiver.

Fig. 2.1 illustrates the proximity based positioning technique. The estimated position of the mobile user M_2 is the position of the base station BS_1 since it is in the coverage area of only BS_1. For the mobile users M_1 and M_3, the receivers can detect signals from more than one base station, so the estimated position is the position of the base station which has the strongest signal strength, e.g., BS_1 for M_1, and BS_2 for M_3.

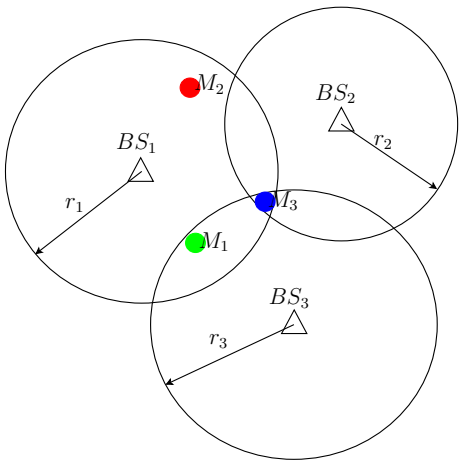


Figure 2.1.: Proximity-based positioning: the estimated position of the mobile user M_i is the position of the base station BS_j if M_i detects only signal from BS_j, or the observed signal strength of BS_j is the strongest among the detected singals

The Active Badge location system [42, 43] is one of the earliest proximity based indoor positioning systems. The system consists of the mobile active badges, each of which periodically generates a unique code (every $15\,\mathrm{s}$), and a network of sensors distributed over the deployment area. Badges and sensors communicate to each other via pulse-width modulated infrared signals. A server (master station) polls the sensors for badge sightings, processes the data, and then determines the position of the badge as the position of the sensor which was last sighted by it.

Recently, a very up-to-date positioning technique using proximity technique is the iBeacon positioning system where the Bluetooth Low Energy (BLE) technology is used [44]. In

an iBeacon positioning system, a number of BLE transmitters which periodically generate advertising data packets which contain the ID of the transmitters and their pre-calibrated transmission power, are installed at the interested positions which are distributed over the deployment area. The optimal broadcast interval is $100\,\mathrm{ms}$ and the configurable broadcast range is from a few centimeters up to hundreds of meters. The compatible mobile devices pick up the data packets from iBeacon transmitters once they are in the broadcast range of the transmitters and make the positioning decisions by processing the observed data packets.

2.2. Lateration

The lateration methods use the absolute range (distance) or the range differences from multiple reference points, i.e., base stations, to compute the position of a mobile object. The distances are computed based on the estimated travelling time of the signal from the transmitter to the receiver, i.e., TOA, TDOA, given the propagation speed. Alternatively the RSSI data with the pre-estimated path loss model is employed to estimate the distances.

Once the distances are obtained, the position of the mobile object can be computed by the following methods:

- Circular lateration:
  These methods estimate the position by solving the system of equation as expressed in Eq. (2.1) using the absolute distances:

$$r_i^2 = (x - x_i)^2 + (y - y_i)^2. \tag{2.1}$$

  In Eq. (2.1), (x, y) and (x_i, y_i) are the coordinates of the mobile object and the i-th reference point in 2-dimentional Cartesian coordinates, respectively, and r_i is the estimated distance between them, where $i = 1, \ldots, N$ and N is the number of observed base stations. Theoretically, when the distances from at least 3 reference points to the mobile object are obtained, it is possible to solve the system of equations to get the unique final estimated coordinates of the mobile object. However, in practic, the distance estimates are often accompanied with errors. Sources of error can be error in time synchronization between the clocks of the transmitter and the receiver, non line of sight (NLOS) problem [45] or variations of the signal strength which are not reflected by the path loss model. These errors may make the system of equations as defined in Eq. (2.1) have no unique solution, i.e., the circles in Fig. 2.2 do not intersect at one unique point. With such situations, an approximation method is needed to solve the system of equations such as the Least Squares (LS) solution. To use LS method, the system of equations can be re-written as follows [3]:

$$\begin{aligned} r_i^2 - r_1^2 &= (x - x_i)^2 + (y - y_i)^2 - (x - x_1)^2 - (y - y_1)^2 \\ &= x_i^2 + y_i^2 - (x_1^2 + y_1^2) - 2x(x_i - x_1) - 2y(y_i - y_1); i = 2, \ldots, N. \end{aligned} \tag{2.2}$$

  The system of equations as defined in Eq. (2.2) can be written in the matrix form:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.3}$$

where

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.4}$$

$$\mathbf{A} = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ \vdots & \vdots \\ x_N - x_1 & y_N - y_1 \end{pmatrix} \tag{2.5}$$

$$\mathbf{b} = \frac{1}{2} \begin{pmatrix} (x_2^2 + y_2^2) - (x_1^2 + y_1^2) - (r_2^2 - r_1^2) \\ \vdots \\ (x_N^2 + y_N^2) - (x_1^2 + y_1^2) - (r_N^2 - r_1^2) \end{pmatrix} \tag{2.6}$$

The LS solution for the above system of equations can be obtained as follows:

$$\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}. \tag{2.7}$$

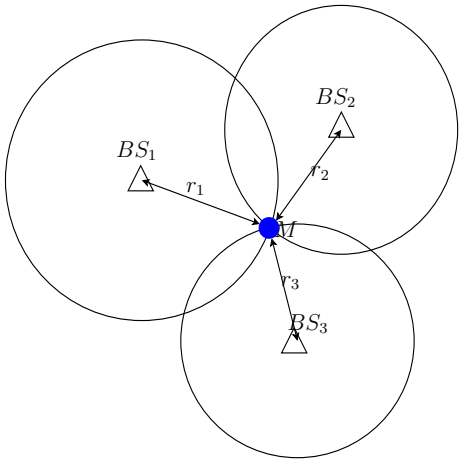


Figure 2.2.: Cicular lateration based positioning: the estimated position of the mobile user M is determined based on the estimated distances between the mobile user and the reference points. At least three reference points are needed to calculate the user position.

- Hyperbolic lateration:
  These methods use range differences between the mobile object and any pair of reference points to form a system of equations of hyperbolas (Eq. (2.8)) which are then used to compute the position of the receiver:

$$\begin{aligned} d_{ij} &= r_i - r_j \\ &= \sqrt{(x - x_i)^2 + (y - y_i)^2} - \sqrt{(x - x_j)^2 + (y - y_j)^2}, \end{aligned} \tag{2.8}$$

  here, d_{ij} denotes the range difference between i-th and j-th reference points, $\forall i, j; i \neq j$.

  The solution can be easily obtained by applying the LS method in a similar fashion as discussed in the circular lateration based method. The above system of equations can

be solved as follows [3]:
Since

$$d_{i1} = r_i - r_1, \tag{2.9}$$

we have

$$\begin{aligned} &(r_1 + d_{i1})^2 = r_i^2 \\ &\Leftrightarrow r_i^2 - r_1^2 = d_{i1}^2 + 2r_1 d_{i1} \end{aligned} \tag{2.10}$$

then

$$\begin{aligned} (x - x_i)^2 + (y - y_i)^2 - (x - x_1)^2 - (y - y_1)^2 &= r_i^2 - r_1^2 \\ &= d_{i1}^2 + 2r_1 d_{i1} \\ \Leftrightarrow x_i^2 + y_i^2 - (x_1^2 + y_1^2) - 2x(x_i - x_1) - 2y(y_i - y_1) &= d_{i1}^2 + 2r_1 d_{i1} \\ x(x_i - x_1) + y(y_i - y_1) + d_{i1} r_1 &= \frac{1}{2}\left((x_i^2 + y_i^2) - (x_1^2 + y_1^2) - d_{i1}^2\right); \end{aligned} \tag{2.11}$$

where $i = 1, \cdots, N$.
The above system of equations in Eq. 2.11 can be written in matrix form:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.12}$$

where

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ r_1 \end{pmatrix} \tag{2.13}$$

$$\mathbf{A} = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 & d_{21} \\ \vdots & \vdots & \\ & \vdots & \\ x_N - x_1 & y_N - y_1 & d_{N1} \end{pmatrix} \tag{2.14}$$

$$\mathbf{b} = \frac{1}{2}\begin{pmatrix} (x_2^2 + y_2^2) - (x_1^2 + y_1^2) - d_{21}^2 \\ \vdots \\ (x_N^2 + y_N^2) - (x_1^2 + y_1^2) - d_{N1}^2 \end{pmatrix} \tag{2.15}$$

The LS solution for the above system of equations is then obtained as:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}. \tag{2.16}$$

Many indoor localization methods employing lateration techniques have been proposed, see the descriptions in [46, 47, 48] and the references therein. The positioning accuracy depends on how accurate the estimate of the distance between transmitter and receiver is. Lateration based techniques can be divided into 2 categories: time based lateration and signal strength based lateration.

Time based lateration techniques employ the TOA or TDOA information to compute the distances. In [49], a time based lateration positioning system has been developed. The object to be located is the mobile tag transmitting ultrasonic pulses in a roughly hemispherical pattern around the top of it, once requested from a master station via a radio signal. A central positioning unit polls the network of the ultrasonic receivers mounted on the ceiling of the room to retrieve the time of flight (TOF) of the ultrasound emitted by the mobile tag (if detected at the receiver). The TOF information is measured as the time difference between the moment the master sends out the request to mobile tags for emitting ultrasonic pulse and the first signal peak detected on the receiver. For time synchronization, the master station sends the reset signal to all receivers simultaneously with the ultrasound emitting request. Distances between mobile tag and receivers are then computed and the position of the mobile tag is determined by applying the LS approach. The reported experimental results are very impressive with a positioning error in the order of centimeters.

In addition to time based lateration, signal strength based lateration techniques are also employed in indoor positioning. These techniques employ the dependence of signal strength on propagation distance. Therefore, the accuracy of these techniques mostly depends on how accurate the estimated path loss model is.

In [18], a modified path loss model for indoor environment was developed, where the attenuation of signal penetrating through walls is considered. The parameters of the path loss model are trained empirically. Once the distances between transmitters and receiver are obtained, the final estimate of receiver location is obtained by various methods, e.g., by solving the system of circular or hypecbolic equations using the LS approach as summarized in [46].

In [47], methods for improving positioning results using RSSI based lateration techniques were explored. Instead of using the theoretical path loss model, the authors proposed two approaches, regression based and correlation based. In the regression based method, polynomial regression was used to model the relationship between the RSSI and distance, and the coefficients of the polynomial are estimated by employing LS approximation. The correlation based method utilizes the fact that the signal propagation from close-by locations to an AP is highly correlated as they face the similar propagation environment. This method recursively performs localization in gradually reduced local areas to approach the true location. In each iteration, a certain number of RSSI readings which are the closest training points to the previous estimated position are kept and used to fit the propagation model. Experimental results on both artificial data and real data showed the considerable improvements compared to the original RSSI based lateration methods.

An overview of lateration based techniques for Ultra-Wideband signals is given in [48] where positioning schemes for time based lateration and RSSI based lateration are discussed.

In general, time based lateration positioning techniques are more precise than RSSI based lateration techniques, however, they require additional hardware.

2.3. Angulation

Angulation based positioning techniques employ the angle of arrival of a wireless signal to determine the position of a mobile object. Theoretically, these techniques are able to compute the position of the receiver when there are at least two reference points, and if the position

of the mobile object does not lie on the line connecting the reference points.

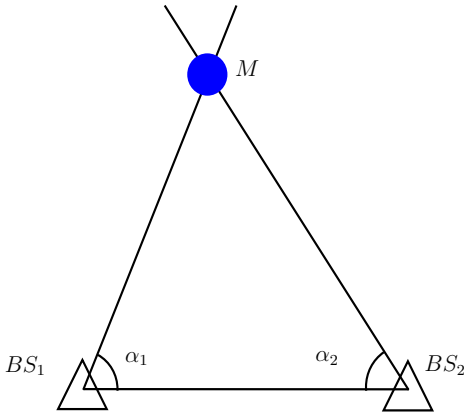


Figure 2.3.: Angulation based positioning: the estimated position of the mobile user M is determined based on the estimated angle α_i between the mobile user and the reference points. At least two reference points are needed to calculate the user position with the condition that the user position does not lie on the line connecting the reference points.

Fig. 2.3 illustrates the angulation based positioning techniques. In this system, the connected line between two references can be considered as an internal reference. The angle between the transmitter and the receiver α_i can be determined by:

$$\begin{aligned} \tan\alpha_i &= \frac{y - y_i}{x - x_i} \\ \Leftrightarrow x\sin\alpha_i - y\cos\alpha_i &= x_i\sin\alpha_i - y_i\cos\alpha_i; i = 1, \cdots, N. \end{aligned} \tag{2.17}$$

However, these techniques also suffer from the errors caused by the NLOS problem. If there are more than two reference points, the system of equations might not produce a unique solution. To solve the system of equations, an approximate solution is needed. Again, the LS method is the most common solution for solving the system of equations which is defined by Eq. (2.17).

Eq. (2.17) can be written in matrix form as follows

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.18}$$

where

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.19}$$

$$\mathbf{A} = \begin{pmatrix} -\sin\alpha_1 & \cos\alpha_1 \\ \vdots & \vdots \\ -\sin\alpha_N & \cos\alpha_N \end{pmatrix} \tag{2.20}$$

$$\mathbf{b} = \begin{pmatrix} y_1\cos\alpha_1 - x_1\sin\alpha_1 \\ \vdots \\ y_N\cos\alpha_N - x_N\sin\alpha_N \end{pmatrix} \tag{2.21}$$

The LS solution for the above system of equations can be obtained as follows:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}. \tag{2.22}$$

An indoor satellite positioning system has been proposed in [38], however, different from the satellite based GPS. This system employs the infrared signals instead of eletromagnetic wave and angulation technique instead of lateration technique. The system consists of 3 infrared light sources, called indoor satellites, mounted at fixed positions as emitters, and the infrared incident angle sensors at the mobile objects measuring the incident angle from each emitter. The position of the mobile objects are then obtained by applying the LS approach.

In [50], angulation based techniques have been employed to estimate the current position of an object. ML, LS, Total Least Squares (TLS) and Weighted LS (WLS) algorithms were applied to solve the system of equations. To improve the performance of the positioning system, a method based on Weighted TLS (WTLS) was derived. The optimistic results with WTLS based approach were presented with simulation data which approach the Cramer-Rao Lower Bound (CRLB).

Another example of an AOA based localization system can be found in [51]. The system consists of a set of passive thermal infrared sensors installed in the room edges to detect the thermal radiation of the human skin. The thermal sensors are thermopile-arrays, where each contains a number of pixels and each pixel has a field of view. The heat source position is determined via the principle of AOA by computing the intersection point created by the directions of the pixels with the highest outputs.

2.4. Fingerprinting

Fingerprinting based positioning techniques are the methods to estimate the position of an object which rely on training data from a set of reference points (anchor points) with known locations. Fig. 2.4 illustrates such a fingerprinting based system.

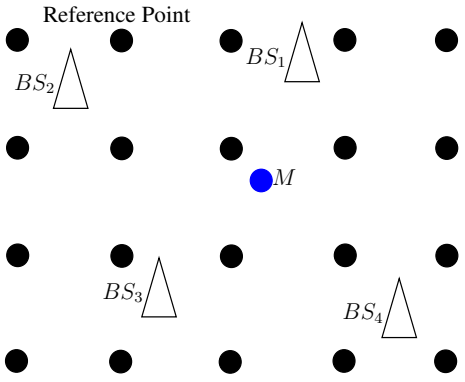


Figure 2.4.: Fingerprinting based positioning: the filled circles indicates the reference (anchor) points, while the triangles show base station locations. The user position is the position of the anchor point whose training data best match the online measurement

Fingerprinting based methods can be well formulated as a machine learning and pattern recognition approach. These techniques generally consist of two phases: training phase and classification phase (also refered as offline phase and online phase). In the training phase,

the training data, i.e., RSSI, are collected at the anchor points and used to build the database which is often called radio map which describes the RSSI-position relationship. The radio map can be defined as follows

$$\mathcal{R} = \{(\ell_k, \mathcal{F}_k) \,|k = 1, \cdots, K\} \tag{2.23}$$

where ℓ_k is the k-th reference position, K is the number of reference positions in the deployment area, $\mathcal{F}_k$ can be either the set of raw measurements, i.e., $\mathcal{F}_k = \mathcal{X}_k = \mathbf{x}_{k,1}, \cdots, \mathbf{x}_{k,N}$, where $\mathbf{x}_{k,n} = [x_{k,n,1}, \cdots x_{k,n,N_{AP}}]^T$ is the n-th measurement vector which contains the RSSIs from N_{AP} base stations, or it contains the class conditional probability density functions (PDFs), i.e., $\mathcal{F}_k = p(\mathbf{x}|\ell_k)$, estimated from the measurements at the k-th position.

During the classification phase, the online measurements are compared against the training data at every anchor point. The position of the anchor point whose training data best match the online data can be considered as the estimated position of the receiver. As discussed in chapter 1, there are two most common approaches for calculating the similarity between training data and online data: deterministic approaches, e.g., the k-nearest neighbor rule, and probabilitic approaches, e.g., the Bayesian classification rule.

Though radio fingerprinting based positioning techniques can be applied to both indoor and outdoor environments, it seems that these techniques are more suitable for indoor since the distinction of radio signal strengths observed at different positions indoor is much higher than outdoor due to the density of obstructions in indoor environments.

In the very well-known fingerprinting based positioning system Radar [18, 21], the k-nearest neighbor method is employed. The position of the receiver is computed by averaging the coordinates of the k anchor points which have highest similarity between the training data and the online observation. The similarity is measured by computing the Euclidian distance between the training data and online observation in signal strength space as follows

$$\mathcal{D}(\mathbf{o}, \mathbf{x}_{k,n}) = \sqrt{\sum_{i=1}^{N_{AP}} (o_i - x_{k,n,i})^2}, \forall k, \forall n \tag{2.24}$$

where $\mathbf{o} = [o_1, \cdots, o_{N_{AP}}]^T$ and $\mathcal{D}$ denote the online observation and the distance, respectively. The j-th position is considered to be the nearest neighbor, i.e., the training data that best match the online observation, if there is a training measurement $\mathbf{x}_{j,m}$ which satisfies

$$\mathcal{D}(\mathbf{o}, \mathbf{x}_{j,m}) \leq \mathcal{D}(\mathbf{o}, \mathbf{x}_{k,n}), \forall k \neq j, \forall n \tag{2.25}$$

In [22], an extensive analysis of the fingerprinting based positioning system that employs the Euclidian distance is presented. The effect of the number of access points, the number of training samples per position, the density of the anchor points, etc. on the performance of the positioning system are analyzed. The analysis results provide a guideline on choosing parameters to design and deploy an indoor positioning system.

In probabilitic approaches, in the training phase the statistical parameters of the PDF of the signal strength are trained. During the classification phase, the similarity between training data and online data is computed by calculating the likelihood of observing the online data given the PDF of signal strength at anchor points [24, 25, 8], or some other criteria to measure the similarity such as the Bhattacharyya distance [52], then again a k-nearest neighbor algorithm can be applied to compute the final estimated position of the receiver.

There are two methods to estimate the PDF of the training data, parametric and non-parametric density estimation techniques [3]. Parametric estimation methods assume a model, e.g., a Gaussian, for the density function and aim to estimate the parameters of the model [25, 8], i.e., $\mathcal{F}_k = \theta_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. On the other hand, non-parametric methods do not assume any prior model but estimate the class conditional PDF by using the histogram method [24, 52] or Kernel density estimators [23, 53].

After constructing the class conditional PDF, the final position estimate can be obtained by computing the likelihood or the posterior using the Bayes' rule. The position which achieves the highest likelihood or posterior is then considered as the position of the mobile receiver. Maximum likelihood estimator aims to find the position that maximizes the probability of observing the online measurement $\mathbf{o}$, given the class conditional PDFs at reference points, as follows

$$\hat{\ell} = \underset{\ell_k}{\operatorname{argmax}}\, p(\mathbf{o}|\ell_k). \tag{2.26}$$

A maximum a posteriori estimator finds the position which has the highest posterior probability as

$$\hat{\ell} = \underset{\ell_k}{\operatorname{argmax}}\, P(\ell_k|\mathbf{o}). \tag{2.27}$$

Applying Bayes' rule, $P(\ell_k|\mathbf{o})$ can be obtained as follows

$$\begin{aligned} P(\ell_k|\mathbf{o}) &= \frac{p(\mathbf{o}|\ell_k)P(\ell_k)}{p(\mathbf{o})} \\ &= \frac{p(\mathbf{o}|\ell_k)P(\ell_k)}{\sum_{k'=1}^{K} p(\mathbf{o}|\ell_{k'})P(\ell_{k'})}, \end{aligned} \tag{2.28}$$

assuming that the prior $P(\ell_k)$ is given.

In [25], two different types of class conditional PDF are considered, one is the parametric model (Gaussian) and the other is the non-parametric model. In the reported results, the method employing the parametric model outperforms the other. This is because, as stated, the parametric technique smooths the distribution shape to account for missing signal strength values in the training phase, (due to the finite number of training samples) which avoids obtaining a zero probability for any signal strength value that was not observed in the training phase.

2.5. Dead Reckoning

Dead reckoning (DR) techniques employ the information from a system of inertial sensors, i.e., acceleration and gyroscope sensors, in order to estimate the position of an object based on the previous estimated position. DR techniques had been first developed for the aviation and marine industry, nowadays they are also used in robotics [54, 55, 56], indoor positioning [57, 58, 59, 32], and many more systems.

Fig. 2.5 illustrates the positioning procedure of DR techniques where the current position is estimated based on the previous estimate. $\mathbf{v}_t$ is the movement vector computed from displacement information and movement heading estimation.

With the development of Microelectromechanical systems (MEMS) technology, ordinary smartphones now often have a built-in inertial measurement unit (IMU) which allows to employ the dead reckoning techniques for indoor pedestrian tracking without additional hardware. However, they have lower accuracy in comparison with the IMU system used on airplanes or in space science.

As discussed in chapter 1, these techniques are often developed in a combination with other methods using other sources of information, such as WiFi signal or GPS, which are used to regularly reset the error of DR techniques which accumulates over time and distance. Displacement estimation of the indoor user in DR based indoor positioning is done via step detection and step length estimation where the observed acceleration data are used. Various methods have been proposed for step detection procedure such as peak detection [57], zero-crossing detection as mentioned in [59] or autocorrelation approach [32]. Step length estimation based on acceleration data is a tricky task since the accuracy of built-in sensors on smartphones are often very low. As a result, most of the step length estimation methods are based on a calibration phase to estimate some heuristic factors, as summarized in [60]. A Kalman filter is often used for movement heading estimation which utilizes the information from gyroscope and magnetic sensors [59].

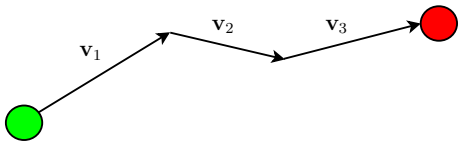


Figure 2.5.: Dead reckoning: the user position is estimated based on the estimated movement vectors assuming that the starting point is given.

2.6. Comparison of Techniques

All the above mentioned radio signal based techniques for positioning have their own advantages and disadvantages and their suitability depends on the deployment environment.

- Proximity based techniques can be applied for any low cost positioning system for either outdoor or indoor (no requirement of additional hardware) which does not require a high positioning accuracy. The positioning error depends on the coverage range of the base stations, e.g., the proximity positioning systems using GSM base stations have error in the order of hundreds of meters. An indoor positioning system may use these techniques employing the wireless local area network (WLAN) system. However the positioning error is about the coverage range of an AP, i.e., roughly $100\,\mathrm{m}$ which is unacceptable in the indoor environment. Moreover, in indoor environments, the contour of signal strength radiated from an AP is obviously not symmetric due to the presence of obstructions. This leads to misclassification results, because the receiver may receive a higher RSSI from a farther AP because of line of sight to this AP while it observes a lower RSSI from a closer AP due to absence of line of sight. Positioning using iBeacon advertising packets is another approach which aims to produce a room precise localization system. However, the higher the accuracy requirements, the more hardware (iBeacon transmitters) needs to be installed.

- Lateration and angulation based techniques are able produce highly precise positioning results in free space environment where signals can propagate by the direct path from the transmitter to the receiver and almost no multipath problem occurs. Unfortunately, the indoor environment is definitely not free space. It has a lot of obstructions such as walls, furniture or movement of people that makes the accuracy of the positioning using these techniques very limited. Because of the special requirements of the environment, these techniques may only be suitable for positioning in a single room. Another limitation of these techniques is the requirement of additional specialized hardware, i.e., antenna, microphone arrays, etc., and deep hardware access which seems to be not possible for a positioning system employing the existing WLAN infrastructure and ordinary mobile devices, i.e., smartphones. Clock synchronization is another challenge for lateration based techniques which determines the accuracy of positioning results.

- Fingerprinting based methods as discussed above consist of two phases, training phase and classification phase. The first and the foremost disadvantage of these techniques is the necessity of a training phase which is very time consuming and need to be done very carefully because the classification results are strongly influenced by the quality of the training data. Another disadvantage of these techniques is that the training data need to be updated regularly in order to adapt to the changes of the infrastructure and environment in order to maintain the accuracy of the positioning results. Despite these disadvantages, fingerprinting based techniques are still the most promising methods for indoor environments for three main reasons: First, they can produce a reasonable positioning result without any additional special hardware. Second, these techniques are suitable to be employed in an already installed WLAN system, and WLAN is available in many places. Third, systems using these techniques can be applied in a large area and have a good scalability.

- Dead reckoning techniques are able to produce precise positioning results given the initial position without any knowledge of the infrastructure in the coverage area, however, only over a short time and movement distance. The high accuracy is not kept long because of the accumulation of error over time and distance of movement. The errors are caused, for example, by the drift or bias of sensor systems, and also the noise in the environments, for example the measurement from a magnetic sensor is strongly affected by metallic materials arround the sensor and not only the magnetic field of the earth.

From the analysis of the the advantages and disadvantages of the mentioned radio signal based techniques, for developing a real indoor positioning system based on the regular smartphones and the already installed WLAN system, the combination of fingerprinting based and dead reckoning techniques appears to be the best solution because of the following reasons:

- **Sufficient accuracy**: The positioning accuracy meets the requirement of indoor positioning for the common purposes such as user tracking and security, in large indoor areas, i.e., universities, museums, hospitals or stores.

- **Cost efficiency**: no additional hardware is needed to build up a positioning system.

- **Deployment simplicity**: WLAN is today available in almost every indoor environment. Consequently, the positioning system can be deployed in many indoor areas easily.
- **Robustness**: The combination of WiFi fingerprinting based and DR techniques makes the system robust, i.e., the mobile devices can do self localization based on DR techniques when no WiFi signal is available.

3. Scientific Objectives

This study aims to investigate methods to enhance the indoor positioning and navigation for smartphone users employing an already deployed WLAN system and the built-in inertial sensors of the smartphones.

First, with WiFi fingerprinting based positioning, a statistical approach in the machine learning and pattern recognition framework is applied. The statistical approach helps to address the variations of the WiFi signal which are obvious in the indoor environment due to the multipath propagation caused by walls, obstructions, movement of people, and so on. Reports of other research show that the positioning accuracy highly depends on how good the training database is. In this study, the training database contains the statistical parameters of the measured WiFi RSSI. Therefore, first the characteristics of the WiFi data are analysed. WiFi data which suffer from the censoring and dropping problems have not been addressed in any previous research. Censoring, i.e., clipping, refers to the fact that sensors are unable to measure RSSI values below some threshold, such as $-100\,\mathrm{dBm}$. Dropping means that occasionally RSSI measurements of access points are not available, although their value is clearly above the censoring threshold. While censoring occurs due to the limited sensitivity of WiFi sensors on portable devices, dropping comes from the limitation of sensor drivers and the operation of WLAN system. Assuming that WiFi data observed from an AP at a position follow a Gaussian distribution, our goal is to develop a parameter estimator which is able to cope with censoring and dropping problems. It is noted that these problems also occur in online measurements, therefore, the other goal of this study is to derive a classification rule which takes into account the censoring and dropping problems. The details of the proposed parameter estimator and classification rule are presented in Chapter 4.

RSSI statistics from different smartphones are not identical when the smartphones are from different vendors because of different hardware and software. Moreover, according to our data investigation, smartphones from the same vendor and even the same model are not able to observe the same signal strength when they measure the data at the same conditions. Though several previous studies have investigated methods to solve this mismatch problem, non of which has considered the presence of censored and dropped data. As a result, another goal of this study is to address the problem of the differences of the sensors sensitivities amongst smartphones when censored and dropped data are present in the measurement data. Chapter 5 discusses our proposed "smartphone adaptation" methods. Furthermore we adopt an adaptation method from the speech recognition field for the first time to smartphone adaptation.

Methods for combining the fingerprinting based and dead reckoning techniques to improve the positioning results using a hidden Markov model (HMM) or a Kalman filter have been explored in several previous research which indicated that better positioning accuracy is obtained in comparison with methods employing any single source of data. Within this work,

HMM is chosen to improve positioning results since the sensor fusion and map information are easily incorporated by an HMM where the HMM states are the positions where fingerprints have been taken. In a large area, training data can only be collected at a fairly coarse grid of states in order to reduce the training effort, which leads to a high quantization error. Therefore, this work aims to deal with the problem of how to achieve a low discretization error in the positioning results without increasing training effort. The proposed methods are presented in detail in Chapter 6.

Another aim of this work is to develop a real positioning and navigation system which is able to operate within the area of the University of Paderborn. The expected system must satisfy the following requirements: low cost, ease of deployment, maintenance and update, user friendliness, robustness and high scalability of deployment area as well as system features. The developed indoor positioning and navigation system is briefly presented in Chapter 8.

4. Parameter Estimation of Censored and Dropped Gaussian Data

4.1. Motivation

As mentioned in previous chapters, fingerprinting based positioning techniques can be formulated as a machine learning and pattern recognition task, where the training phase corresponds to the learning and the localization/classification phase is the pattern recognition procedure. Localization results do not only depend on the localization procedure but also depend highly on how well the models are trained.

The discussion in the previous chapters has stated that in fingerprinting based positioning, probabilistic approaches perform better than deterministic approaches. Moreover, as the discussion in [25, 61] showed, the class conditional PDF can be parametric or non-parametric, and the former seems to outperform the latter. There are several discussions about the PDF of WiFi data. In [62, 61], researchers stated that the WiFi RSSI data follow a Gaussian while the study in [63] shows that the distribution is not symmetric, potentially left-skewed. According to our investigations, data of most of the readings follow Gaussian distributions if a sufficient amount of samples is collected. However, we also observed some readings which follow a skewed distribution. Fortunately, the skewed tail is not much longer than the tail of a normal PDF resulting in the similarity of the mode, the median and the mean of skewed and normal distribution. Therefore, Gaussian distribution is chosen as the model for WiFi data throughout this work.

According to our data investigation, we have detected two problems of WiFi RSSI data, censoring and dropping, which strongly affect the parameter estimates, and consequently, the positioning results. While the censoring problem is obvious because of the limited sensitivity of the WiFi chipset, the dropping problem is not easy to be recognized. The data investigation which points out the dropping problem is presented in [64]. The censoring and dropping problems are illustrated in Fig. 4.1 which shows histograms obtained from field data.

Intuitively, Fig. 4.1(a) seems to show the histogram of data drawn from a Gaussian. For this case, parameters of the Gaussian are easily obtained by using ML estimation. However, most of the readings belong to one of the three latter cases which are shown in Fig. 4.1(b), (c) and (d) where the unobservable measurements are set to -100 dBm.

What does the histogram in Fig. 4.1(b) tell us? As can be seen, the left side part of the Gaussian is missing which would be complemented by the unobservable measurements (were assigned the value of -100 dBm) to make the Gaussian complete. This situation is meant when we state that censored data are present in the measurements. Censoring occurs due to the limited sensitivity of WiFi sensors or the sensor driver which does not report intentio-

nally the too weak observed signal strengths, i.e., the smartphones do not report the signal strength if it is less than a certain threshold c, e.g., in our data set $c = -100\,\mathrm{dBm}$.

Fig. 4.1(c) shows the histogram of data drawn from a complete, uncensored Gaussian. However, there are still unobservable measurements reported as can be seen by the histogram bar at $-100\,\mathrm{dBm}$. In this situation, these unobservable measurements are supposedly caused by the dropping problem. Dropped data are the unobservable measurements of an AP which is supposedly close enough to the measurement position that the smartphone should always observe signal transmitted from that AP above the threshold c. Dropping may occur due to the limitation of the WiFi chipset driver, i.e., limited buffer sizes or time-outs, and the operation of the WLAN system, i.e., switch off of AP.

Finally, in Fig. 4.1(d), one part of Gaussian is missing which is similar to Fig. 4.1(b). However, as can be seen in the figure, the amount of unobservable measurements seem to be much larger than the missing part of the Gaussian. This phenomenon can only be explained when in such situation, WiFi data experiences both censoring and dropping.

It has to be noted that, the unobservable data due to censoring and dropping describe different problems compared to the problem of not observing data for some specific values due to limited measurement time as discussed in [25]. Therefore, if the measurement time is infinite, the problem stated in [25] is solved. However, censoring and dropping problems remain unsolved. Obviously, traditional histogram method for class conditional PDF estimation is not able to address the missing data problems since they rely on the observable training data only. This leads to a severe limitation of non-parametric model that the likelihood of the unobservable measurements, i.e., censored and dropped data, cannot be computed correctly, i.e., the likelihoods computed from unobservable data are always zero. This again motivated us to use the parametric model since all the limitations of non-parametric approaches can be handled efficiently. As a consequence, an estimator which is able to estimate the parameters of censored and dropped Gaussian data was developed which improves the WiFi fingerprinting based indoor positioning results.

In this chapter, a parameter estimator employing EM algorithm is developed to estimate parameters of censored Gaussian data. Then an extension of the proposed EM algorithm is made to cope with both censored and dropped Gaussian data. For classification, an optimal classification rule which takes into account the presence of censored and dropped data is derived.

For estimating parameters of censored Gaussian data, while ML estimates have been presented in [65], EM algorithm has been derived in [66]. The proposed EM algorithm to estimate the parameters of censored Gaussian data is built upon the result from [66]. Furthermore, this study shows that in case of a single Gaussian model, the estimation algorithm is biasfree and achieves the Cramer-Rao Lower Bound (CRLB) for both mean and variance estimates. An extension of the proposed EM algorithm to cope with dropped data is new, none of the previous research has taken into account this dropping problem.

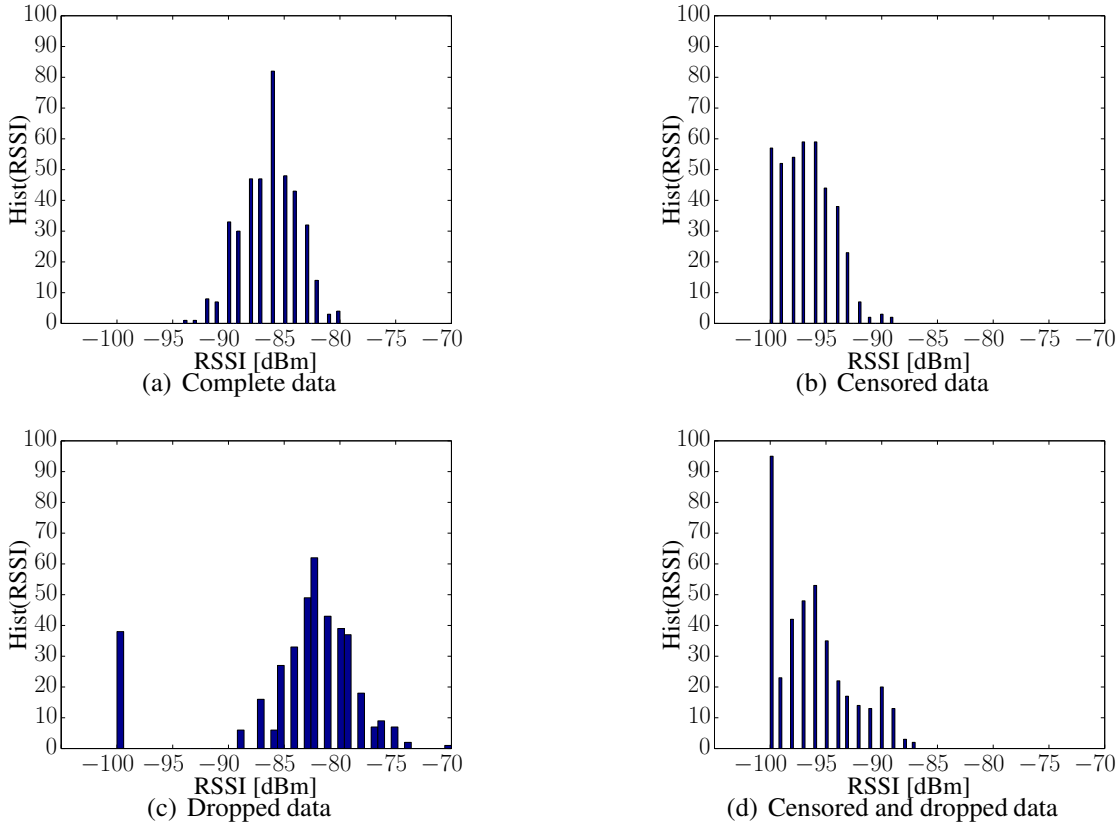


Figure 4.1.: Histogram of real field data illustrates censoring and dropping problem of WiFi data

4.2. Parameter Estimation using EM Algorithm

4.2.1. Introduction to the EM Algorithm

The EM algorithm is an iterative method for ML estimation of parameters of statistical models in the presence of hidden variables. It was first introduced in [67]. This method can be used to estimate the parameters of a Gaussian Mixture Model (GMM).

In a GMM, the PDF of an observation $\mathbf{x}$ is:

$$p(\mathbf{x};\Theta) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x};\theta_k) \tag{4.1}$$

where π_k are positive mixing weights summing to one, p_k is the k-th component density function with the parameters θ_k, and $\Theta = (\pi_1, \cdots, \pi_K, \theta_1, \cdots, \theta_K)$ is the set of all model parameters. Each observation is assumed to be from one of the K components. Assuming that the component density functions are multivariate Gaussians, the parameters θ_k are the set of $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

Let $\mathcal{X} = (\mathbf{x_1}, \ldots, \mathbf{x_N})$ be the set of observable data and $\mathcal{Z} = (z_1, \ldots, z_N)$ denote the hidden variables. Then the log-likelihood function becomes:

$$\ln(p(\mathcal{X};\Theta)) = \ln\left\{\sum_{\mathcal{Z}} p(\mathcal{X},\mathcal{Z};\Theta)\right\}. \tag{4.2}$$

ML estimation can be applied to Eq. (4.2) to obtain the parameter estimates. However, it is often extremely difficult to calculate the summation. To simplify the expression, the complete data are defined to be $\{\mathcal{X}, \mathcal{Z}\}$. Then instead of computing the likelihood of observations, the likelihood of the complete data is computed.

Since the hidden variables are not observable, instead of computing the likelihood directly, the expected value of the log-likelihood of the complete data given the observations and the old estimated parameters are calculated:

$$\begin{aligned} Q(\Theta, \Theta^{(\kappa)}) &= \mathrm{E}\left[\ln\left(p(\mathcal{X}, \mathcal{Z}; \Theta)\right) | \mathcal{X}; \Theta^{(\kappa)}\right] \\ &= \sum_{\mathcal{Z}} \ln\left(p(\mathcal{X}, \mathcal{Z}; \Theta)\right) P(\mathcal{Z}|\mathcal{X}; \Theta^{(\kappa)}) \cdot \end{aligned} \tag{4.3}$$

Here, $\Theta^{(\kappa)}$ denotes the current estimate of the parameters and $Q(\Theta, \Theta^{(\kappa)})$ is called auxiliary function. The calculation of Q is the Expectation step (E-step) of the EM algorithm. Finding the new parameter $\Theta = \Theta^{(\kappa+1)}$ which maximize the auxiliary function is considered as the Maximization step (M-step) of the algorithm. The EM algorithm operates E-step and M-step alternatingly until convergence, i.e., until the improvement of the log-likelihood is smaller than a threshold. In [67], the authors have proved that the improvement of $Q(\Theta, \Theta^{(\kappa)})$ will improve the log-likelihood of observable data. However, there is no guarantee that the results will converge to a ML estimator, i.e., an EM algorithm may converge to a local maximum of the observable data likelihood function only.

4.2.2. EM algorithm for Censored Gaussian Data

In this section, an EM algorithm for estimating parameters of the univariate single Gaussian data in the presence of censored data is presented. The censoring problem can be regarded as shown in Fig. 4.2:

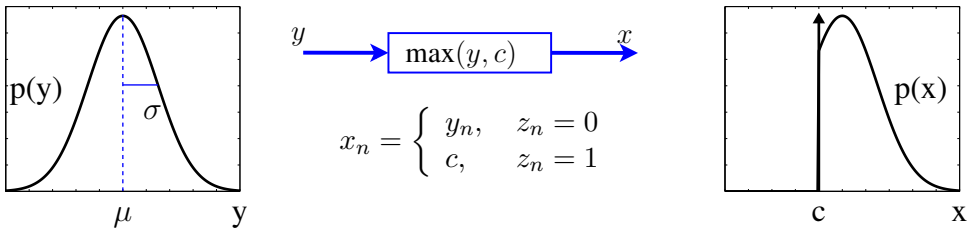


Figure 4.2.: Censoring problem: left figure: PDF from where y is drawn; right figure: PDF from where the observation x is drawn.

In the following derivation we consider that Gaussian measurements are one-sided censored only to simplify the exposition. An extension to the two-sided case is straightforward.

Notations

Let $\mathbf{y} = y_1, ..., y_N; y_n \in \mathbb{R}$ be the unobservable, non-censored data, where N is the number of measurements and where the y_n are i.i.d. with Gaussian probability density function (PDF) $p_Y(y_n) = \mathcal{N}(y_n; \mu, \sigma^2)$. Observable are the data $\mathbf{x} = x_1, ..., x_N$, where $x_n = \max(y_n, c)$, with clipping threshold c. Throughout this dissertation, the words "clip" and "censor" are

used with the same meaning. The goal is to estimate the parameters $\theta = \{\mu, \sigma^2\}$ of the underlying Gaussian.

E-Step

Employing the EM algorithm we identify $\mathbf{y}$ and $\mathbf{x}$ to be the complete and the observed data, respectively. Thus the expected log-likelihood of the complete data is given by

$$Q(\theta;\theta^{(\kappa)}) = \mathrm{E}\left[\ln\left(p_Y(\mathbf{y};\theta)\right)|\mathbf{x};\theta^{(\kappa)}\right] \tag{4.4}$$

$$= \sum_{n=1}^{N}\int_{-\infty}^{\infty}\ln\left(p_Y(y_n;\theta)\right)p\left(y_n|x_n;\theta^{(\kappa)}\right)\mathrm{d}y_n \tag{4.5}$$

$$=: \sum_{n=1}^{N} f_n\left(\theta;\theta^{(\kappa)}\right), \tag{4.6}$$

where $f_n\left(\theta;\theta^{(\kappa)}\right) = \int_{-\infty}^{\infty}\ln\left(p_Y(y_n;\theta)\right)p\left(y_n|x_n;\theta^{(\kappa)}\right)\mathrm{d}y_n$ and where κ is the iteration index.

The term $p(y_n|x_n;\theta^{(\kappa)})$ can be determined as follows:

$$p(y_n|x_n;\theta^{(\kappa)}) = \begin{cases} \delta(y_n - x_n) & \text{, if } x_n > c \\ \frac{p(x_n|y_n;\theta^{(\kappa)})p(y_n;\theta^{(\kappa)})}{p(x_n;\theta^{(\kappa)})} & \text{, if } x_n = c \end{cases} \tag{4.7}$$

For the case $x_n = c$ we know that $y_n \leq c$. $p(y_n|x_n;\theta^{(\kappa)})$ can thus be calculated as follows

$$\begin{aligned} p(y_n|x_n;\theta^{(\kappa)}) &= \frac{p(x_n|y_n;\theta^{(\kappa)})p(y_n;\theta^{(\kappa)})}{p(x_n;\theta^{(\kappa)})} \\ &= \frac{p(x_n|y_n;\theta^{(\kappa)})p(y_n;\theta^{(\kappa)})}{\int_{-\infty}^{c}p(x_n|y_n;\theta^{(\kappa)})p(y_n;\theta^{(\kappa)})\mathrm{d}y_n} \\ &= \frac{\delta(x_n - c)p(y_n;\theta^{(\kappa)})}{\int_{-\infty}^{c}\delta(x_n - c)p(y_n;\theta^{(\kappa)})\mathrm{d}y_n} \\ &= \frac{\mathcal{N}\left(y_n;\theta^{(\kappa)}\right)}{I_0(\theta^{(\kappa)})} \end{aligned}$$

Here we have used the notation

$$I_j(\theta^{(\kappa)}) = \int_{-\infty}^{c} y^j\mathcal{N}\left(y;\theta^{(\kappa)}\right)\mathrm{d}y \tag{4.8}$$

with $j = 0$. $I_0(\theta^{(\kappa)})$ can be calulated as follows (see Appendix A.1.1 for the detailed computation):

$$I_0(\theta^{(\kappa)}) = \frac{1}{2}\mathrm{erfc}\left(-\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right). \tag{4.9}$$

Then we obtain

$$p(y_n|x_n;\theta^{(\kappa)}) = \begin{cases} \delta\left(y_n - x_n\right) & \text{, if } x_n > c \\ \frac{\mathcal{N}\left(y_n;\theta^{(\kappa)}\right)}{I_0(\theta^{(\kappa)})} & \text{, if } x_n = c \end{cases}. \tag{4.10}$$

Introducing the binary random variable Z_n with realization z_n, where $z_n = 0$ and $z_n = 1$ indicate that the n-th measurement is not censored or censored, respectively, the summand in Eq. (4.6) can be written as

$$f_n\left(\theta;\theta^{(\kappa)}\right) = \frac{z_n}{I_0\left(\theta^{(\kappa)}\right)}\int_{-\infty}^{c}\ln\left(\mathcal{N}(y_n;\theta)\right)\mathcal{N}\left(y_n;\theta^{(\kappa)}\right)\mathrm{d}y_n + (1-z_n)\ln\left(\mathcal{N}(x_n;\theta)\right). \tag{4.11}$$

here, the fact that $y_n = x_n$ when $z_n = 0$ is exploited.

M-Step

The parameter re-estimation formulas are obtained by computing the derivatives of Eq. (4.11) w.r.t. the elements of θ and set them to zero:

$$\begin{aligned}\frac{\partial}{\partial\mu}f_n\left(\theta;\theta^{\kappa}\right) &= \frac{z_n}{I_0\left(\theta^{(\kappa)}\right)}\int_{-\infty}^{c}\frac{y_n-\mu}{\sigma^2}\mathcal{N}\left(y_n;\theta^{(\kappa)}\right)\mathrm{d}y_n \\ &\quad + (1-z_n)\frac{x_n-\mu}{\sigma^2} \\ &= \frac{z_n}{\sigma^2}\frac{I_1\left(\theta^{(\kappa)}\right)-\mu I_0\left(\theta^{(\kappa)}\right)}{I_0\left(\theta^{(\kappa)}\right)} + (x_n-\mu)\frac{1-z_n}{\sigma^2} \qquad (4.12)\\ \frac{\partial}{\partial\sigma}f_n\left(\theta;\theta^{\kappa}\right) &= \frac{z_n}{I_0\left(\theta^{(\kappa)}\right)}\int_{-\infty}^{c}\left(-\frac{1}{\sigma}+\frac{(y_n-\mu)^2}{\sigma^3}\right)\mathcal{N}\left(y_n;\theta^{(\kappa)}\right)\mathrm{d}y_n \\ &\quad + (1-z_n)\left(-\frac{1}{\sigma}+\frac{(x_n-\mu)^2}{\sigma^3}\right) \\ &= \frac{z_n}{\sigma}\left[\frac{1}{\sigma^2}\left(\frac{I_2(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}-2\mu\frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}+\mu^2\right)-1\right] \\ &\quad + \frac{1-z_n}{\sigma}\left(\frac{(x_n-\mu)^2}{\sigma^2}-1\right), \qquad (4.13)\end{aligned}$$

Here, we have used the notation as defined in Eq. (4.8). Setting the sumations of the above derivatives to zeros, re-estimation formulas are readily obtained:

$$\begin{aligned}\sum_{n=1}^{N}\frac{\partial}{\partial\mu}f_n\left(\theta;\theta^{\kappa}\right)\Big|_{\mu=\mu^{\kappa+1}} &= 0: \\ \mu^{(\kappa+1)} &= \frac{1}{N}\frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}\sum_{i=1}^{N}z_n + \frac{1}{N}\sum_{i=1}^{N}(1-z_n)x_n \qquad (4.14)\\ \sum_{n=1}^{N}\frac{\partial}{\partial\sigma}f_n\left(\theta;\theta^{\kappa}\right)\Big|_{\sigma^2=(\sigma^2)^{\kappa+1}} &= 0: \\ \left(\sigma^2\right)^{(\kappa+1)} &= \left[\frac{I_2(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}-2\mu^{(\kappa)}\frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}+\left(\mu^2\right)^{(\kappa)}\right]\frac{1}{N}\sum_{i=1}^{N}z_n \\ &\quad + \frac{1}{N}\sum_{i=1}^{N}(1-z_n)\left(x_n-\mu^{(\kappa)}\right)^2. \qquad (4.15)\end{aligned}$$

The computation of $I_1(\theta^{(\kappa)})$ and $I_2(\theta^{(\kappa)})$ are given in Appendix A.1.2 and Appendix A.1.3, respectively.

As can be seen in Eq. (4.14) and Eq. (4.15), the unobservable data, i.e., censored data, also contribute to the estimates beside the observable ones. Intuitively, in case of absence of censored data, i.e., $z_n = 0$ for all measurements, the re-estimation formulas reduce to the typical ML estimation formulas for mean μ and variance σ^2.

To have a better intuition of the re-estimation formulas, the situation after convergence of the estimates is considered. Let $\mu^{(\kappa+1)} \approx \mu^{(\kappa)} =: \hat{\mu}$ and $(\sigma^2)^{(\kappa+1)} \approx (\sigma^2)^{(\kappa)} =: \hat{\sigma}^2$, using this in Eq. (4.14) and Eq. (4.15) and solving for the estimates, we arrive at

$$\hat{\mu} = \frac{M}{N}\frac{1}{M}\sum_{i=1}^{M} x_n + \left[1 - \frac{M}{N}\right] \frac{\int_{-\infty}^{c} y p_Y(y;\hat{\theta})\mathrm{d}y}{\int_{-\infty}^{c} p_Y(y;\hat{\theta})\mathrm{d}y}, \tag{4.16}$$

$$\hat{\sigma}^2 = \frac{M}{N}\frac{1}{M}\sum_{i=1}^{M} (x_n - \hat{\mu})^2 + \left[1 - \frac{M}{N}\right] \frac{\int_{-\infty}^{c} (y-\hat{\mu})^2 p_Y(y;\hat{\theta})\mathrm{d}y}{\int_{-\infty}^{c} p_Y(y;\hat{\theta})\mathrm{d}y}, \tag{4.17}$$

where we assumed without loss of generality that the first $M = \sum_i (1 - z_n)$ observations are the uncensored ones. These expressions lend themselves to the following interpretation: Mean and variance estimates are the weighted average between their ML estimates which are computed from the observed data and the mean and variance of the assumed truncated Gaussian of the unobservable parts. The weights are the relative frequencies of the uncensored and censored measurements, respectively.

Properties of the Estimates

In the following, the main properties of the proposed estimator are investigated. As will be shown, the proposed EM algorithm delivers virtually **biasfree** and **efficient** estimates.

Unbiasness and Convergence

In order to study the convergence properties the expected values of the difference between the estimates, Eqs. (4.14) and (4.15), and the true values of the parameters are computed. First note that Z_n is a Bernoulli random variable with

$$P(Z_n = 1) = \int_{-\infty}^{c} p_Y(y;\theta)\mathrm{d}y = I_0(\theta)$$

and $P(Z_n = 0) = 1 - I_0(\theta)$. Thus: $\mathrm{E}[Z_n] = \mathrm{E}[Z_n^2] = I_0(\theta)$.

Taking the expectation of mean estimate Eq. (4.14) we have

$$\mathrm{E}\left[\mu^{(\kappa+1)}\right] = \frac{1}{N}\sum_{i=1}^{N} \mathrm{E}[(1 - Z_n)X_n] + \frac{1}{N}\sum_{i=1}^{N} \mathrm{E}\left[\frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})} Z_n\right]. \tag{4.18}$$

It has to be noted that the estimate $\theta^{(\kappa)}$ depends on the data and is thus random. Thus the expectation operator also applies to this quantity. The first expectation can be evaluated as follows:

$$\begin{aligned}\mathrm{E}[(1 - Z_n)X_n] &= \sum_{z_n=0}^{1} \int_{-\infty}^{\infty} (1 - z_n) x_n P(z_n|x_n) p_Y(x_n) dx_n \\ &= \int_{c}^{\infty} x_n \mathcal{N}(x_n;\theta) dx_n = \mu - I_1(\theta).\end{aligned} \tag{4.19}$$

Note that here we have used the notation $p_Y(x_n)$ because $x = y$ in case of not censored. Here we have employed the fact that

$$P(z_n = 0|x_n) = \begin{cases} 1, & \text{if } x_n > c \\ 0, & \text{else} \end{cases} . \tag{4.20}$$

Using further $\mathrm{E}\left[\frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})} z_n\right] \approx \mathrm{E}\left[\frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}\right] \mathrm{E}\left[Z_n\right]$ and subtracting μ from either side of Eq. (4.18) we obtain

$$\mathrm{E}\left[\tilde{\mu}^{(\kappa+1)}\right] = -I_1(\theta) + I_0(\theta)\mathrm{E}\left[\frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}\right]. \tag{4.21}$$

where $\tilde{\mu}^{(\kappa+1)} = \mu^{(\kappa+1)} - \mu$.

In order to compute the remaining expectation, a Taylor series expansion around the true parameter values $\theta = (\mu, \sigma^2)$ is applied and truncated after the linear term:

$$\mathrm{E}\left[\frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}\right] \approx \frac{I_1(\theta)}{I_0(\theta)} + \mathrm{E}\left[\tilde{\mu}^{(\kappa)}\right] \frac{\partial}{\partial\mu}\frac{I_1(\theta)}{I_0(\theta)} + \mathrm{E}\left[\left(\tilde{\sigma}^2\right)^{(\kappa)}\right] \frac{\partial}{\partial\sigma^2}\frac{I_1(\theta)}{I_0(\theta)}. \tag{4.22}$$

Expectation of variance estimate Eq. (4.15) can be calulated in a similar procedure as follows:

$$\begin{aligned}
\mathrm{E}\left[\sigma^{2(\kappa+1)}\right] &= \mathrm{E}\left[\frac{1}{N}\sum_{n=1}^{N}(1 - z_n)(x_n - \mu)^2\right] \\
&\quad + \mathrm{E}\left[\frac{1}{N}\sum_{n=1}^{N} z_n \frac{I_2(\theta^{(\kappa)}) - 2I_1(\theta^{(\kappa)})\mu + I_0(\theta^{(\kappa)})\mu^2}{I_0(\theta^{(\kappa)})}\right] \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathrm{E}\left[(1 - z_n)(x_n - \mu)^2\right] \\
&\quad + \frac{1}{N}\sum_{n=1}^{N}\mathrm{E}\left[z_n \frac{I_2(\theta^{(\kappa)}) - 2I_1(\theta^{(\kappa)})\mu + I_0(\theta^{(\kappa)})\mu^2}{I_0(\theta^{(\kappa)})}\right]
\end{aligned} \tag{4.23}$$

The first expectation can be evaluated as follows:

$$\begin{aligned}
\mathrm{E}\left[(1 - z_n)(x_n - \mu)^2\right] &= \sum_{z_n=0}^{1}\int_{-\infty}^{\infty}(1 - z_n)(x_n - \mu)^2 P(z_n|x_n)p_Y(x_n)dx_n \\
&= \int_{c}^{\infty}(x_n - \mu)^2\mathcal{N}(x_n;\theta)dx_n \\
&= \sigma^2 - (I_2(\theta) - 2\mu I_1(\theta) + \mu^2 I_0(\theta)).
\end{aligned} \tag{4.24}$$

Using further $\mathrm{E}\left[z_n \frac{I_2(\theta^{(\kappa)}) - 2I_1(\theta^{(\kappa)})\mu + I_0(\theta^{(\kappa)})\mu^2}{I_0(\theta^{(\kappa)})}\right] \approx \mathrm{E}\left[\frac{I_2(\theta^{(\kappa)}) - 2I_1(\theta^{(\kappa)})\mu + I_0(\theta^{(\kappa)})\mu^2}{I_0(\theta^{(\kappa)})}\right] \mathrm{E}\left[Z_n\right]$ and subtracting σ^2 from either side of Eq. (4.23) we obtain

$$\begin{aligned}
\mathrm{E}\left[\left(\tilde{\sigma}^2\right)^{(\kappa+1)}\right] &= -\left(I_2(\theta) - 2\mu I_1(\theta) + \mu^2 I_0(\theta)\right) \\
&\quad + I_0(\theta)\mathrm{E}\left[\frac{I_2(\theta^{(\kappa)}) - 2I_1(\theta^{(\kappa)})\mu + I_0(\theta^{(\kappa)})\mu^2}{I_0(\theta^{(\kappa)})}\right]
\end{aligned} \tag{4.25}$$

where $(\tilde{\sigma}^2)^{(\kappa+1)} = (\sigma^2)^{(\kappa+1)} - \sigma^2$.

In order to compute the remaining expectation, a Taylor series expansion around the true parameter values $\theta = (\mu, \sigma^2)$ is again applied and truncated after the linear term:

$$\begin{aligned}\mathrm{E}\left[\frac{I_2(\theta^{(\kappa)}) - 2I_1(\theta^{(\kappa)})\mu + I_0(\theta^{(\kappa)})\mu^2}{I_0(\theta^{(\kappa)})}\right] \approx \frac{I_2(\theta) - 2I_1(\theta)\mu + I_0(\theta)\mu^2}{I_0(\theta)} \\ + \mathrm{E}\left[\tilde{\mu}^{(\kappa)}\right]\frac{\partial}{\partial\mu}\frac{I_2(\theta) - 2I_1(\theta)\mu + I_0(\theta)\mu^2}{I_0(\theta)} \\ + \mathrm{E}\left[\left(\tilde{\sigma}^2\right)^{(\kappa)}\right]\frac{\partial}{\partial\sigma^2}\frac{I_2(\theta) - 2I_1(\theta)\mu + I_0(\theta)\mu^2}{I_0(\theta)}.\end{aligned} \tag{4.26}$$

Writing Eq. (4.21) and Eq. (4.25) in matrix form, we arrive at

$$\begin{aligned}\begin{pmatrix}\mathrm{E}\left[\tilde{\mu}^{(\kappa+1)}\right] \\ \mathrm{E}\left[(\tilde{\sigma}^2)^{(\kappa+1)}\right]\end{pmatrix} &\approx \begin{pmatrix}W_\mu & W_{\mu\sigma} \\ W_{\sigma\mu} & W_\sigma\end{pmatrix}\begin{pmatrix}\mathrm{E}\left[\tilde{\mu}^{(\kappa)}\right] \\ \mathrm{E}\left[(\tilde{\sigma}^2)^{(\kappa)}\right]\end{pmatrix} \\ &= \begin{pmatrix}W_\mu & W_{\mu\sigma} \\ W_{\sigma\mu} & W_\sigma\end{pmatrix}^{\kappa+1}\begin{pmatrix}\mathrm{E}\left[\tilde{\mu}^{(0)}\right] \\ \mathrm{E}\left[(\tilde{\sigma}^2)^{(0)}\right]\end{pmatrix},\end{aligned} \tag{4.27}$$

where

$$\begin{aligned}W_\mu &= I_0(\theta)\frac{\partial}{\partial\mu}\frac{I_1(\mu)}{I_0(\mu)}, \quad W_{\mu\sigma} = I_0(\theta)\frac{\partial}{\partial\sigma^2}\frac{I_1(\theta)}{I_0(\theta)}, \\ W_\sigma &= I_0(\theta)\frac{\partial}{\partial\sigma^2}\frac{I_2(\theta) - 2I_1(\theta)\mu + I_0(\theta)\mu^2}{I_0(\theta)}, \\ W_{\sigma\mu} &= I_0(\theta)\frac{\partial}{\partial\mu}\frac{I_2(\theta) - 2I_1(\theta)\mu + I_0(\theta)\mu^2}{I_0(\theta)}.\end{aligned}$$

The W entries can be readily computed by using the derivatives given in Appendix A.1.4.

Investigating all the possible values of the magnitudes of the eigenvalues of the matrix with the W entries by changing the clipping threshold c from $-\infty$ to ∞, we observed that they are always less than one (approaching one for $c \to \infty$, i.e., no observable data). We thus conclude that the estimates are biasfree because the expectation of the error decreases exponentially over iterations. Since, however, in practice μ has to be replaced by its estimate in the estimation of the variance, we exhibit the same bias as in ordinary ML estimation of the variance. In the application considered here, the bias of the ML estimate of the variance can be neglected due to the large number of samples. Further, an estimate of the convergence speed of the EM algorithms can also be obtained from Eq. (4.27), see Fig. 4.3. It can be seen that the number of iterations quickly rises once more than 50% of the data are clipped.

Precision

In order to evaluate the precision of our proposed estimator, we calculated the Cramer-Rao Lower Bound (CRLB) for the estimator. CRLB of the estimation error variance of the mean and variance estimates can be obtained by inverting the information matrix $\mathbf{I}$ which contains the expected values of the second-order partial derivatives of the log-likelihood function.

From the derivation of the EM algorithm it can be seen that the measurements consist of two types of data, the number M of noncensored observations and the observations themselves. While the former type is binomially distributed, the latter type is drawn from a truncated

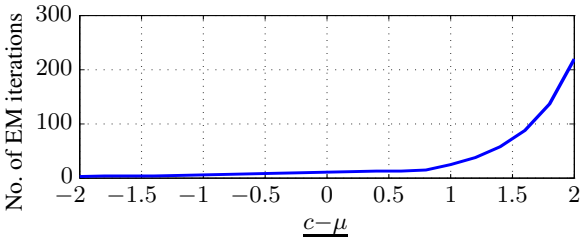


Figure 4.3.: Theoretical number of EM iterations κ required to reduce the estimation error to 10^{-4} of its initial value as a function of $(c-\mu)/\sigma$. Initial values $\mu^{(0)}$, $\left(\sigma^2\right)^{(0)}$ have been set to the ML estimates of μ, σ^2 computed from the unclipped observations only.

Gaussian. It is noted that the draws from the Gaussian are independent of the binomial random variable. Following [65] the likelihood is thus given by, where $I_0(\theta)$ is the probability of not observing a sample and $\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x_j-\mu}{\sigma}\right)^2\right)$ is the probability of observing $x_j > -100\,\mathrm{dBm}$

$$p(\mathbf{x};\theta) = \frac{N!}{M!(N-M)!} I_0^{N-M}(\theta) \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^M \exp\left(-\frac{1}{2}\sum_{j=1}^{M}\left(\frac{x_j-\mu}{\sigma}\right)^2\right), \tag{4.28}$$

and the log-likelihood function is obtained as follows:

$$\begin{aligned}\ln p(\mathbf{x};\theta) = \ln\left(\frac{N!}{M!(N-M)!}\right) + (N-M)\ln\left(I_0(\theta)\right) \\ - \frac{M}{2}\ln\left(2\pi\sigma^2\right) - \frac{1}{2}\sum_{j=1}^{M}\left(\frac{x_j-\mu}{\sigma}\right)^2.\end{aligned} \tag{4.29}$$

The information matrix $\mathbf{I}$ is defined as:

$$\mathbf{I} = \begin{pmatrix} \mathrm{E}\left[-\frac{\partial^2}{\partial\mu^2}\ln p(\mathbf{x};\theta)\right] & \mathrm{E}\left[-\frac{\partial^2}{\partial\mu\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] \\ \mathrm{E}\left[-\frac{\partial^2}{\partial\mu\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] & \mathrm{E}\left[-\frac{\partial^2}{\partial\sigma^2\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] \end{pmatrix}. \tag{4.30}$$

The entries of matrix $\mathbf{I}$ can be calculated as follows (see Appendix A.1.5 for the detailed computation):

$$\mathrm{E}\left[-\frac{\partial^2}{\partial\mu^2}\ln p(\mathbf{x};\theta)\right] = \frac{N}{\sigma^2}\left(\frac{I_1^2 - I_2 I_0}{I_0} + 1\right) \tag{4.31}$$

$$\mathrm{E}\left[-\frac{\partial^2}{\partial\mu\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] = \frac{N}{\sigma^2}\left(-\frac{\frac{\partial I_1}{\partial\sigma^2}I_0 - I_1\frac{\partial I_0}{\partial\sigma^2}}{I_0}\right) \tag{4.32}$$

$$\mathrm{E}\left[-\frac{\partial^2}{\partial\sigma^2\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] = \frac{N}{\sigma^2}\left(\frac{1}{2\sigma^2} - \frac{1}{2\sigma^2}\left(\frac{\frac{\partial I_2}{\partial\sigma^2}I_0 - I_2\frac{\partial I_0}{\partial\sigma^2}}{I_0} - 2\mu\frac{\frac{\partial I_1}{\partial\sigma^2}I_0 - I_1\frac{\partial I_0}{\partial\sigma^2}}{I_0}\right)\right) \tag{4.33}$$

where the computation of the derivative terms are given in Appendix A.1.4.

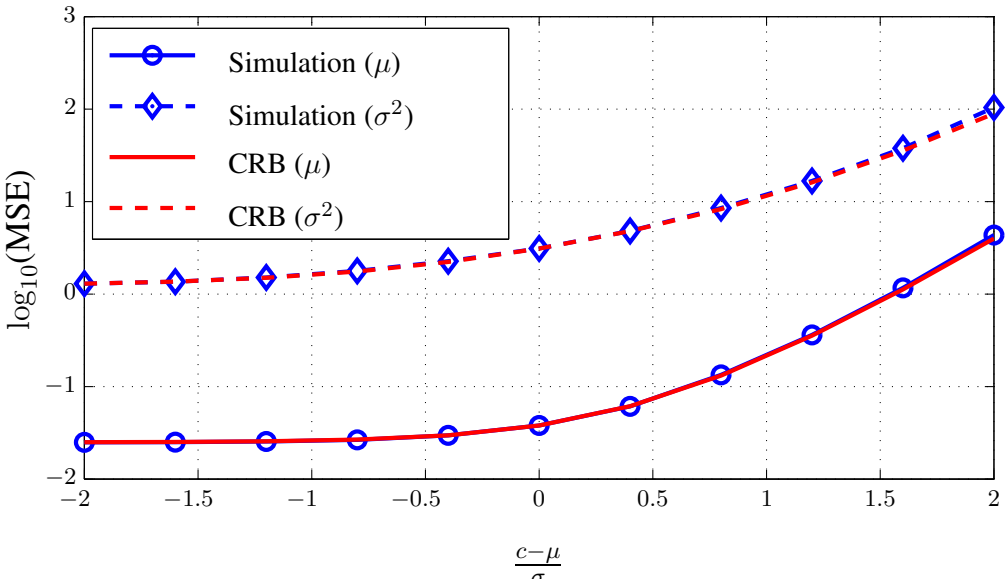


Figure 4.4.: Comparison of CRLB for mean and variance with MSE obtained from simulation for $\sigma^2 = 25$ and $N = 1000$.

Computing the $\mathbf{I}^{-1}$, we obtain CRLB on estimation error variance of μ and σ^2:

$$\text{Var}(\hat{\mu}) \geq \mathbf{I}^{-1}(1,1) \tag{4.34}$$

$$\text{Var}(\hat{\sigma}^2) \geq \mathbf{I}^{-1}(2,2) \tag{4.35}$$

Fig. 4.4 compares the CRLB with the mean squared error (MSE) of the proposed estimators for mean μ and variance σ^2 obtained from a simulation. It can be seen that the estimator practically achieves the bound. The differences are so small that they are not visible in the graph. We therefore conclude that the estimator is efficient. Furthermore, for the limiting case of completely uncensored data the well-known results for ML parameter estimation from a normal population are obtained: $\text{MSE}(\mu) = \sigma^2/N$; $\text{MSE}(\sigma^2) = \sigma^4(2N-1)/N^2$.

4.2.3. EM algorithm for Censored and Dropped Gaussian Data

In this subsection, the proposed EM algorithm for parameter estimation of censored Gaussian data is extended to be able to cope not only with the censored data but also the dropped data.

Notations

The measurement model for censored and dropped data is shown in Fig. 4.5. Here, the hidden variables d_n, $n = 1, \ldots N$, indicate whether an observation is dropped ($d_n = 1$) or not ($d_n = 0$), where $P(d_n = 1)$ is the *dropping rate*, in the following, $P(d_n = 1)$ is denoted by π. The variables are gathered in the binary vector $\mathbf{d} = [d_1, \ldots, d_N]$, where N is the number of measurements. Furthermore, define $\mathbf{y} = y_1, ..., y_N$, where the y_n are i.i.d. with Gaussian probability density function (PDF) $p_Y(y_n) = \mathcal{N}(y_n; \mu, \sigma^2)$ if $d_n = 0$, and $y_n = c$ if $d_n = 1$, where we set c to the smallest measurable RSSI value (e.g., $c = -100\,\text{dBm}$). It should be

noted that $\mathbf{y}$ is not the complete data anymore since the data are possibly dropped. Observable data are the censored, possibly dropped data $\mathbf{x} = x_1, ..., x_N$, where $x_n = \max(y_n, c)$, with the censoring threshold c as above.

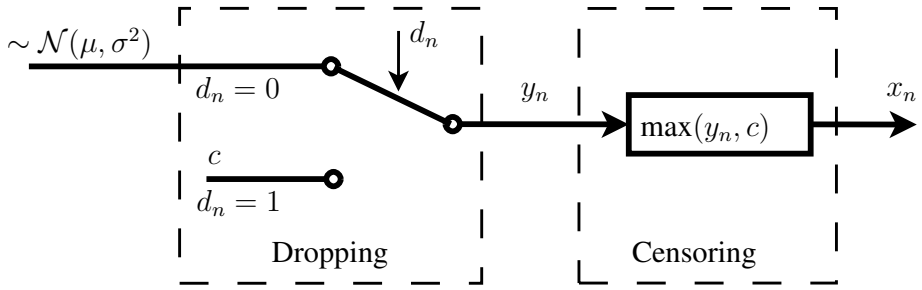


Figure 4.5.: Measurement model.

The difference between censoring and dropping can be expressed as follows: Even if the parameters of the Gaussian are such that practically no censoring occurs (because $\mu \gg c$), dropping will still possibly occur.

The goal is to estimate the parameters $\theta = \{\mu, \sigma^2, \pi\}$. This will be achieved by the Expectation Maximization (EM) algorithm, where the complete data are $\{\mathbf{y}, \mathbf{d}\}$ and the observable are $\{\mathbf{x}\}$.

E-Step

It is noted that x_n does not convey more information about θ than y_n. So, the complete data are $\{\mathbf{y}, \mathbf{d}\}$ rather than $\{\mathbf{x}, \mathbf{y}, \mathbf{d}\}$. The expected log-likelihood of the complete data, given the observed one, is given by:

$$\begin{aligned} Q(\theta; \theta^{(\kappa)}) &= \mathrm{E}\left[\ln\left(p(\mathbf{y}, \mathbf{d}; \theta)\right) | \mathbf{x}; \theta^{(\kappa)}\right] \\ &= \sum_{n=1}^{N} \sum_{d_n=0}^{1} \int_{-\infty}^{\infty} \ln\left(p(y_n, d_n; \theta)\right) p(y_n, d_n | x_n; \theta^{(\kappa)}) \mathrm{d}y_n, \end{aligned} \tag{4.36}$$

where κ is the iteration index.

$p(y_n, d_n|x_n)$ can be written as

$$p(y_n, d_n|x_n; \theta^{(\kappa)}) = p(y_n|d_n, x_n; \theta^{(\kappa)}) P(d_n|x_n; \theta^{(\kappa)}). \tag{4.37}$$

For calculating $p(y_n|d_n, x_n; \theta^{(\kappa)})$, two cases are considered:

- For data that are not dropped ($d_n = 0$), see Eq. (4.10),

$$p(y_n|d_n = 0, x_n; \theta^{(\kappa)}) = \begin{cases} \delta(y_n - x_n), & \text{if } x_n > c \\ \frac{\mathcal{N}(y_n; \theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}, & \text{if } x_n = c \end{cases} \tag{4.38}$$

- For data that are dropped ($d_n = 1$)

$$p(y_n|d_n = 1, x_n; \theta^{(\kappa)}) = \begin{cases} 0, & \text{if } x_n > c \\ \delta(y_n - c), & \text{if } x_n = c \end{cases}. \tag{4.39}$$

Furthermore, the posterior of the hidden variable d_n can be computed using Bayes' rule

$$P(d_n|x_n;\theta^{(\kappa)}) = \frac{p(x_n|d_n;\theta^{(\kappa)})P(d_n)}{\sum_{d_n=0}^{1} p(x_n|d_n;\theta^{(\kappa)})P(d_n)}. \tag{4.40}$$

Since there are two variables, four cases can be discerned. Using the notation $\beta_n(d,z) := P(d_n|z_n;\theta^{(\kappa)})$, where $z_n = 1$ indicates that $x_n = c$ and $z_n = 0$ that $x_n > c$, we obtain:

$$\begin{aligned}\beta_n(1,1) &= P(d_n = 1|x_n = c;\theta^{(\kappa)}) \\ &= \frac{p(x_n = c|d_n = 1)P(d_n = 1)}{p(x_n = c|d_n = 0)P(d_n = 0) + p(x_n = c|d_n = 1)P(d_n = 1)} \\ &= \frac{\pi^{(\kappa)}}{I_0(\theta^{(\kappa)})(1-\pi^{(\kappa)}) + \pi^{(\kappa)}},\end{aligned} \tag{4.41}$$

where the result from the previous section is employed: $p(x_n = c|d_n = 0) = I_0(\theta^{(\kappa)})$ is the probability that a measurement is censored, i.e., unobservable, given that it is not dropped. In addition, $p(x_n = c|d_n = 1)$ is always equal to 1 since if the measurement is dropped, it is assigned the value of c. $\beta_n(1,1)$ is the probability that the measurement is dropped given it is unobservable.

The probability that the measurement is not dropped given it is unobservable, $\beta_n(0,1)$, is then easily obtained since $\beta_n(0,1) + \beta_n(1,1) = 1$.

Furthermore, it is obvious that if a measurement is observable, it cannot be dropped, so $\beta_n(1,0) = 0$, and thus $\beta_n(0,0) = 1$.

We further note that

$$\begin{aligned}p(y_n, d_n;\theta) &= P(d_n;\theta)p(y_n|d_n;\theta) \\ &= \begin{cases} \pi\delta(y_n - c), & \text{if } d_n = 1 \\ (1-\pi)\mathcal{N}(y_n;\theta), & \text{if } d_n = 0 \end{cases}\end{aligned} \tag{4.42}$$

Using all the above in (4.36) we arrive at

$$\begin{aligned}Q(\theta;\theta^{(\kappa)}) = \sum_{n=1}^{N} \Bigg\{ & z_n \int_{-\infty}^{c} \ln\left((1-\pi)\mathcal{N}(y_n;\theta)\right) \frac{\mathcal{N}(y_n;\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})} \beta_n(0,1)\mathrm{d}y_n \\ & + (1-z_n)\int_{c}^{\infty} \ln\left((1-\pi)\mathcal{N}(y_n;\theta)\right)\delta(y_n - x_n)\beta_n(0,0)\mathrm{d}y_n \\ & + z_n \int_{-\infty}^{c} \ln\left(\pi\delta(y_n - c)\right)\delta(y_n - c)\beta_n(1,1)\mathrm{d}y_n \Bigg\} \\ = \sum_{n=1}^{N} \Bigg\{ & z_n \ln(1-\pi)\beta_n(0,1) \\ & + \frac{z_n}{I_0(\theta^{(\kappa)})} \int_{-\infty}^{c} \ln\left(\mathcal{N}(y_n;\theta)\right)\mathcal{N}(y_n;\theta^{(\kappa)})\mathrm{d}y_n\beta_n(0,1) \\ & + (1-z_n)\ln(1-\pi) + (1-z_n)\ln\left(\mathcal{N}(x_n;\theta)\right) \\ & + z_n \ln(\pi)\beta_n(1,1) \Bigg\}.\end{aligned} \tag{4.43}$$

M-Step

Computing the derivative of the auxiliary function Eq. (4.36) the following iterative parameter estimation formulas can be readily derived:

$$\mu^{(\kappa+1)} = \frac{\sum_{n=1}^{N}(1-z_n)x_n + \frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})}\sum_{n=1}^{N} z_n\beta_n(0,1)}{N - \sum_{n=1}^{N} z_n\beta_n(1,1)} \tag{4.44}$$

$$\begin{aligned}\left(\sigma^2\right)^{(\kappa+1)} = &\frac{\sum_{n=1}^{N}\left[z_n\left(\frac{I_2(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})} - 2\mu\frac{I_1(\theta^{(\kappa)})}{I_0(\theta^{(\kappa)})} + \mu^2\right)\beta_n(0,1)\right]}{N - \sum_{n=1}^{N} z_n\beta_n(1,1)} \\ &+ \frac{\sum_{n=1}^{N}\left[(1-z_n)(x_n-\mu)^2\right]}{N - \sum_{n=1}^{N} z_n\beta_n(1,1)}\end{aligned} \tag{4.45}$$

$$\pi^{(\kappa+1)} = \frac{\sum_{n=1}^{N} z_n\beta_n(1,1)}{N}. \tag{4.46}$$

These formulas reduce to the re-estimation formulas for censored data presented in the previous section if the dropping rate is set to $\pi = 0$. Then $\beta_n(1,1) = 0$ and $\beta_n(0,1) = 1$.

As can be seen in Eqs. (4.44), (4.45) and (4.46), not only observable and censored data, but also dropped data contribute to the estimates.

4.3. Optimal Classification Rule for Censored and Dropped Gaussian Data

Above ML parameter estimation is done for all possible user locations ℓ_k and it is done for the measurements of each AP seperately, assuming that data from different APs are independent. The final parameter estimates are denoted by $\hat{\boldsymbol{\mu}}_k = [\hat{\mu}_{k,1}, \cdots, \hat{\mu}_{k,N_{AP}}]^T$ and $\hat{\boldsymbol{\Sigma}}_k = \text{diag}\left[\hat{\sigma}^2_{k,1}, \cdots, \hat{\sigma}^2_{k,N_{AP}}\right]$, where N_{AP} is number of observable APs.

Indoor localization can be formulated as a classification problem, where the classes are the positions from which RSSI measurements are taken during the offline training phase. For each position ℓ_k the parameters of a Gaussian class-conditional density $p_Y(\mathbf{x}|\ell_k)$ of RSSI measurements are estimated using the EM algorithm of the last sections. During online classification, to estimate the user's location, an optimal classification rule is developed.

Let $\mathbf{x} = x_1, \cdots, x_{N_{AP}}$ be the vector of the online measurement. First the posterior is calculated as follows

$$p(\ell_k|\mathbf{x}) = \frac{\prod_{i=1}^{N_{AP}} p(x_i|\ell_k)P(\ell_k)}{\sum_{k'=1}^{K}\prod_{i=1}^{N_{AP}} p(x_i|\ell_{k'})P(\ell_{k'})} \tag{4.47}$$

where K is the number of offline training locations and x_i is the RSSI of i-th AP. $P(\ell_k)$ is the prior on the position. Here we have employed the assumption of the independence of the RSSIs of different APs. It is noted that the test data are also subject to censoring.

The likelihood $p(x_i|\ell_k)$ in Eq. (4.47) can be calculated as follows for censored Gaussian data

$$p(x_i|\ell_k) = \int_{-\infty}^{\infty} p(x_i|y_i)p_Y(y_i|\ell_k)\mathrm{d}y_i, \tag{4.48}$$

where

$$p(x_i|y_i) = \begin{cases} \delta(x_i - y_i), & \text{if } x_i > c \\ \delta(x_i - c), & \text{if } x_i = c \end{cases} . \tag{4.49}$$

Using Eq. (4.49) in Eq. (4.48) we arrive at

$$p(x_i|\ell_k) = \begin{cases} \mathcal{N}(x_i; \hat{\mu}_{k,i}, \hat{\sigma}^2_{k,i}), & \text{if } x_i > c \\ I_0(\hat{\mu}_{k,i}, \hat{\sigma}^2_{k,i}), & \text{if } x_i = c \end{cases} . \tag{4.50}$$

For censored and dropped Gaussian data, $p(x_i|\ell_k)$ in Eq. (4.47) can be determined as follows

$$\begin{aligned} p(x_i|\ell_k) &= \int_{-\infty}^{\infty} \sum_{d_i=0}^{1} p(x_i, y_i, d_i; \hat{\theta}_{k,i}) \mathrm{d}y_i \\ &= \int_{-\infty}^{\infty} p(x_i, y_i, d_i = 0; \hat{\theta}_{k,i}) \mathrm{d}y_i + \int_{-\infty}^{\infty} p(x_i, y_i, d_i = 1; \hat{\theta}_{k,i}) \mathrm{d}y_i \\ &= \int_{-\infty}^{\infty} p(x_i|y_i, d_i = 0; \hat{\theta}_{k,i}) p(y_i|d_i = 0; \hat{\theta}_{k,i}) P(d_i = 0; \hat{\theta}_{k,i}) \mathrm{d}y_i \\ &\quad + \int_{-\infty}^{\infty} p(x_i|y_i, d_i = 1; \hat{\theta}_{k,i}) p(y_i|d_i = 1; \hat{\theta}_{k,i}) P(d_i = 1; \hat{\theta}_{k,i}) \mathrm{d}y_i. \end{aligned} \tag{4.51}$$

If $y_i > c$, obviously no dropping problem occurs, consequently $P(d_i = 1; \hat{\theta}_{k,i}) = 0$ and $p(x_i|y_i, d_i = 0; \hat{\theta}_{k,i}) = \delta(x_i - y_i)$. On the other hand, if $y_i \leq c$, the term in Eq. (4.51) can be calculated as follows

$$\begin{aligned} p(x_i|y_i, d_i = 0; \hat{\theta}_{k,i}) &= \delta(x_i - c) \\ P(d_i = 0; \hat{\theta}_{k,i}) &= 1 - \hat{\pi}_{k,i} \\ p(x_i|y_i, d_i = 1; \hat{\theta}_{k,i}) &= \delta(x_i - c) \\ p(y_i|d_i = 1; \hat{\theta}_{k,i}) &= \delta(y_i - c) \\ P(d_i = 1; \hat{\theta}_{k,i}) &= \hat{\pi}_{k,i} \end{aligned}$$

Using the above expressions in Eq. (4.51) we obtain

$$p(x_i|\ell_k) = \begin{cases} \mathcal{N}(x_i; \hat{\mu}_{k,i}, \hat{\sigma}^2_{k,i}), & \text{if } x_i > c \\ (1 - \hat{\pi}_{k,i}) I_0(\hat{\mu}_{k,i}, \hat{\sigma}^2_{k,i}) + \hat{\pi}_{k,i}, & \text{if } x_n = c \end{cases} . \tag{4.52}$$

In Eqs. (4.50) and (4.52), $(\hat{\mu}_{k,i}, \hat{\sigma}^2_{k,i}, \hat{\pi}_{k,i})$ are the estimated parameters of the i-th AP at location ℓ_k. In case all observations of the i-th AP at location ℓ_k are clipped, the mean estimate is set to a small value $\hat{\mu}_{\ell_k,i} \ll c$ and $\hat{\sigma}^2_{\ell_k,i}$ is set to an average value.

Using the set P of nearest neighbors which is chosen among the offline locations by taking those with the largest posteriors, the final location estimate $\hat{\ell}$ is then obtained by the weighted average

$$\hat{\ell}(\mathbf{x}) = \frac{1}{\sum_{k \in P} p(\ell_k|\mathbf{x})} \sum_{k \in P} \ell_k p(\ell_k|\mathbf{x}). \tag{4.53}$$

5. Smartphone Adaptation within the MLLR Framework

5.1. Motivation

Obviously, an Indoor Positioning System (IPS) can only be applied in the real world if it is able to deliver reasonable positioning results to many users with many different smartphones. We acknowledge that until now, most of the research in indoor positioning has been done assuming the device in the localization phase is identical to the one which was used in the training phase. However, this is not realistic because each vendor has its own hardware and software for its product. Consequently, devices may measure different values, although the underlying true value of the quantity to be measured is the same. In case of RSSI measurements, this is due to different antennas and WiFi chipsets. These differences in returned measurements have even been observed with different samples of the same product/smartphone model. This phenomenon is caused by the fact that it is impossible to produce two identical devices because of the error during manufacturing procedure and also the differences in the components, i.e., chipsets and antennas. In [26], the authors reported that the differences of RSSI readings are as high as $-30\,\mathrm{dBm}$ between devices of different vendors and up to $-20\,\mathrm{dBm}$ among devices of the same vendor.

This leads to the problem that the positioning accuracy is dramatically reduced when the test devices are different from the ones used in training. Obviously, it is not feasible to solve the problem by building a database for each device because of the high cost of the training effort. As a result, a question pops up: how can the database which was built upon the measurements from one smartphone be used to localize other smartphones with a reasonable positioning accuracy? This problem is similar to the problem in automatic speech recognition when the database is trained by one speaker and is used to recognize the speech of the others. To solve the problem, a very successful approach to speaker adaptation which is known as Maximum Likelihood Linear Regression (MLLR) was developed. In this chapter, to solve the problem of the different sensitivities of the portable devices, an adaptation method was developed within the MLLR framework.

As discussed in previous chapters, WiFi data suffer from the censoring and dropping problem. Therefore, the proposed adaptation method was developed to be able to adapt training models with adaptation data in the presence of censored and dropped data.

5.2. Model Adaptation in the Presence of Censored and Dropped Data

MLLR adaptation aims at determining the parameters of the adaptation matrices such that the likelihood of the adaptation data is maximized [68, 69]. MLLR can be performed in supervised or unsupervised procedure. In the unsupervised method, the occupying states of adaptation data, i.e., the positions where the adaptation data have been measured, are unknown, whereas in supervised method, the occupying states of adaptation data are known. Since the WiFi signal varies greatly due to many factors, estimating the positions where the observations come from is very challenging, and poor estimation of occupying positions may result in a poor performance of adaptation procedure in the unsupervised case. Within this work, to investigate the effectiveness of adaptation approach, supervised adaptation is used. Though the supervised approach pose additional requirements in the collection of the adaptation data, it still requires much less effort than collecting a completely new set of training data.

According to [70], mean and variance adaptation can be performed in two seperate stages. First, new means are estimated and then, given these new means, the variances are determined. This can be repeated until the changes in means and variances between iterations are smaller than a threshold. As mentioned in [70], the iterative method does not improve results much. However, considerable improvements have been identified within this work when applying iterative method. The reason could be that there is no censored and dropped data in the database which was used in [70]. When censored and dropped data are present in the data, a certain number of iterations is needed since the contribution of those unobservable data to the estimates cannot be estimated correctly after a few iterations, except for the case of no censored and dropped data occur.

5.2.1. Mean Adaptation

In this subsection, the procedure to adapt the trained means with adaptation data in the presence of clipped and dropped data is presented.

Let $\mu_{k,i}$ be the mean value of the Gaussian describing the PDF of the RSSI values of the i-th AP at position ℓ_k. We gather the mean values of all APs in vectors as follows

$$\boldsymbol{\mu}_k = [\mu_{k,1}, \ldots, \mu_{k,i}, \ldots, \mu_{k,N_{AP}}]^T \tag{5.1}$$

for all positions ℓ_k, $k = 1, \ldots, K$.

The goal is to estimate adapted mean vectors $\hat{\boldsymbol{\mu}}_k$ via

$$\hat{\boldsymbol{\mu}}_k = \mathbf{W}^{(c(k))}\boldsymbol{\xi}_k = \mathbf{A}^{(c(k))}\boldsymbol{\mu}_k + \mathbf{b}^{(c(k))}, \tag{5.2}$$

where $\mathbf{W} = [\mathbf{A}|\mathbf{b}]$ is the $(N_{AP} \times (N_{AP}+1))$-dimensional augmented transformation matrix which gathers the parameters of the affine transform, and $\boldsymbol{\xi}_k = (\boldsymbol{\mu}_k^T, 1)^T$ is the extended mean vector. Here, the general case of multiple transformation matrices is considered, where $c(k) \in \{1, C\}$ is the regression class index, which indicates which of the C affine transforms is to be applied to the Gaussian describing position ℓ_k.

To determine the maximum likelihood estimates of the transformation matrix $\mathbf{W}^{(c(k))}$, an EM algorithm is employed which updates $\mathbf{W}^{(c(k))}$ row-wise. Let $\gamma_k(n)$ be the posterior

probability that RSSI measurement vector $\mathbf{x}_n$ is from position ℓ_k. In the case of supervised adaptation that is being considered here it is equal to one if the measurement is indeed from position ℓ_k and zero else.

Let $\mathcal{Y} = \mathbf{y}_1, \cdots, \mathbf{y}_N$, where N is the number of adaptation measurements and $\mathbf{y}_n = y_{n,1}, \cdots, y_{n,N_{AP}}$ is the vector of the measured data of all APs in one measurement. Let $\mathcal{D} = \mathbf{d}_1, \cdots, \mathbf{d}_N$, where $\mathbf{d}_n = d_{n,1}, \cdots, d_{n,N_{AP}}$, $d_{n,i}$ is the random variable which indicates whether the measured data from the i-th AP in the n-th measurement is dropped or not. Observable are censored, possibly dropped data denoted by $\mathcal{X} = \mathbf{x}_1, \cdots, \mathbf{x}_N$, where $\mathbf{x}_n = x_{n,1}, \cdots, x_{n,N_{AP}}$ and $x_{n,i} = \max(y_{n,i}, c)$.

The expected log-likelihood of the complete data $\{\mathcal{Y}, \mathcal{D}\}$, given the observed $\{\mathcal{X}\}$, has a similar form as Eq. (4.36):

$$Q(\lambda; \lambda^{(\kappa)}) = \sum_{k=1}^{K} \sum_{n=1}^{N} \gamma_k(n) \sum_{i=1}^{N_{AP}} \sum_{d_{n,i}=0}^{1} \int_{-\infty}^{\infty} \ln\left(p(y_{n,i}, d_{n,i}; \lambda)\right) p(y_{n,i}, d_{n,i} | x_{n,i}; \lambda^{(\kappa)}) \mathrm{d}y_{n,i} \tag{5.3}$$

where κ is iteration index. Here $\lambda = \{\lambda_{k,i}; k = 1, \ldots, K; i = 1, \ldots, N_{AP}\}$, where $\lambda_{k,i} = \{\pi_{k,i}, \mathbf{w}_i^{(c(k))}, \theta_{k,i}\}$ with $\theta_{k,i} = (\mu_{k,i}, \sigma_{k,i}^2)$, is a short hand notation for the parameters to be estimated. Here, $\mathbf{w}_i^{(c(k))}$ is the row vector which is the i-th row of $\mathbf{W}^{(c(k))}$. In Eq. (5.3) the first sum is over the locations, the second enumerates the adaptation data, while the third is over all APs and the fourth is over the possible values of the random variable $d_{n,i}$.

The terms in Eq. (5.3) can be determined in a similar fashion as the terms in subsection 4.2.3 with the only difference that $y_{n,i}$ is now assumed to be drawn from a Gaussian with adapted parameters: $y_{n,i} \sim \mathcal{N}(\hat{\mu}_{k,i} = \mathbf{w}_i^{(c(k))} \boldsymbol{\xi}_k, \sigma_{k,i}^2)$ as follows. Note that only means are adapted in this stage

$$p(y_{n,i}, d_{n,i}; \lambda_{k,i}) = \begin{cases} \pi_{k,i} \delta(y_{n,i} - c) & \text{, if } d_{n,i} = 1 \\ (1 - \pi_{k,i}) \mathcal{N}(y_{n,i}; \lambda_{k,i}), & \text{, if } d_{n,i} = 0 \end{cases} \tag{5.4}$$

$$p(y_{n,i}, d_{n,i} | x_{n,i}; \lambda_{k,i}^{(\kappa)}) = p(y_{n,i} | d_{n,i}, x_{n,i}; \lambda_{k,i}^{(\kappa)}) P(d_{n,i} | x_{n,i}; \lambda_{k,i}^{(\kappa)}) \tag{5.5}$$

where

$$p(y_{n,i} | d_{n,i} = 0, x_{n,i}; \lambda_{k,i}^{(\kappa)}) = \begin{cases} \frac{\mathcal{N}(y_{n,i}; \lambda_{k,i}^{(\kappa)})}{I_0(\lambda_{k,i}^{(\kappa)})} & \text{, if } x_{n,i} = c \\ \delta(y_{n,i} - x_{n,i}) & \text{, if } x_{n,i} > 0 \end{cases} \tag{5.6}$$

$$p(y_{n,i} | d_{n,i} = 1, x_{n,i}; \lambda_{k,i}^{(\kappa)}) = \begin{cases} \delta(y_{n,i} - c) & \text{, if } x_{n,i} = c \\ 0 & \text{, if } x_{n,i} > 0 \end{cases} . \tag{5.7}$$

Using the notation $\beta_{k,i}(d_{n,i}, z_{n,i}) := P(d_{n,i} | z_{n,i}; \lambda_{k,i}^{(\kappa)})$, where $z_{n,i}$ indicates whether the n-th measured data of the i-th AP is observed $z_{n,i} = 0$ or not $z_{n,i} = 1$. $\beta_{k,i}(d_{n,i}, z_{n,i})$ are then determined as follows

$$\beta_{k,i}(0, 0) = 1 \tag{5.8}$$

$$\beta_{k,i}(1, 0) = 0 \tag{5.9}$$

$$\beta_{k,i}(1, 1) = \frac{\pi_{k,i}^{(\kappa)}}{(1 - \pi_{k,i}^{(\kappa)}) I_0(\lambda_{k,i}^{(\kappa)}) + \pi_{k,i}^{(\kappa)}} \tag{5.10}$$

$$\beta_{k,i}(0, 1) = 1 - \beta_{k,i}(1, 1) \tag{5.11}$$

After using the determined terms as summarized above in Eq. (5.3) we arrive at:

$$
\begin{aligned}
Q(\lambda;\lambda^{(\kappa)}) = \sum_{k=1}^{K}\sum_{n=1}^{N}\gamma_k(n)\sum_{i=1}^{N_{AP}}\Bigg\{ & \\
& z_{n,i}\int_{-\infty}^{c}\ln\left((1-\pi_{k,i})\mathcal{N}(y_{n,i};\lambda_{k,i})\right)\frac{\mathcal{N}(y_{n,i};\lambda_{k,i}^{(\kappa)})}{I_0(\lambda_{k,i}^{(\kappa)})}\beta_{k,i}(0,1)\mathrm{d}y_{n,i} \\
& +(1-z_{n,i})\int_{c}^{\infty}\ln\left((1-\pi_{k,i})\mathcal{N}(y_{n,i};\lambda_{k,i})\right)\delta(y_{n,i}-x_{n,i})\beta_{k,i}(0,0)\mathrm{d}y_{n,i} \\
& +z_{n,i}\int_{-\infty}^{c}\ln\left(\pi_{k,i}\delta(y_{n,i}-c)\right)\delta(y_{n,i}-c)\beta_{k,i}(1,1)\mathrm{d}y_{n,i}\Bigg\} \\
= \sum_{k=1}^{K}\sum_{n=1}^{N}\gamma_k(n)\sum_{i=1}^{N_{AP}}\Bigg\{ & z_{n,i}\beta_{k,i}(0,1)\ln(1-\pi_{k,i}) \\
& +\frac{z_{n,i}\beta_{k,i}(0,1)}{I_0(\lambda_{k,i}^{(\kappa)})}\int_{-\infty}^{c}\ln\left(\mathcal{N}(y_{n,i};\lambda_{k,i})\right)\mathcal{N}(y_{n,i};\lambda_{k,i}^{(\kappa)})\mathrm{d}y_{n,i} \\
& +(1-z_{n,i})\ln(1-\pi_{k,i})+(1-z_{n,i})\ln\left(\mathcal{N}(x_{n,i};\lambda_{k,i})\right) \\
& +z_{n,i}\beta_{k,i}(1,1)\ln(\pi_{k,i})\Bigg\}
\end{aligned}
\tag{5.12}
$$

The goal now is to maximize the objective function $Q(\lambda;\lambda^{(\kappa)})$ w.r.t λ. Using $y_{n,i} \sim \mathcal{N}(\hat{\mu}_{k,i} = \mathbf{w}_i^{(c(k))}\boldsymbol{\xi}_k, \sigma_{k,i}^2)$, Eq. (5.12) can be written as follows

$$
\begin{aligned}
Q(\lambda;\lambda^{(\kappa)}) = \sum_{k=1}^{K}\sum_{n=1}^{N}\gamma_k(n)\sum_{i=1}^{N_{AP}}\Bigg\{ & z_{n,i}\beta_{k,i}(0,1)\ln(1-\pi_{k,i}) \\
& +\frac{z_{n,i}\beta_{k,i}(0,1)}{I_0(\lambda_{k,i}^{(\kappa)})}\int_{-\infty}^{c}\ln\left(\frac{1}{\sqrt{2\pi\sigma_{k,i}^2}}\exp\left(-\frac{\left(y_{n,i}-\mathbf{w}_i^{(c(k))}\boldsymbol{\xi}_k\right)^2}{2\sigma_{k,i}^2}\right)\right)\mathcal{N}(y_{n,i};\lambda_{k,i}^{(\kappa)})\mathrm{d}y_{n,i} \\
& +(1-z_{n,i})\ln(1-\pi_{k,i})+(1-z_{n,i})\ln\left(\frac{1}{\sqrt{2\pi\sigma_{k,i}^2}}\exp\left(-\frac{\left(x_{n,i}-\mathbf{w}_i^{(c(k))}\boldsymbol{\xi}_k\right)^2}{2\sigma_{k,i}^2}\right)\right) \\
& +z_{n,i}\beta_{k,i}(1,1)\ln(\pi_{k,i})\Bigg\}
\end{aligned}
\tag{5.13}
$$

Re-estimation formulas for $\pi_{k,i}$, $\mathbf{w}_i^{(c(k))}$ are obtained by computing the derivatives of the auxiliary function in Eq. (5.13) w.r.t. these parameters and setting them to zero.

Computing derivative of the auxiliary function w.r.t. $\pi_{k,i}$ we arrive at

$$
\begin{aligned}
\frac{\partial}{\partial\pi_{k,i}}Q(\lambda;\lambda^{(\kappa)}) = \sum_{n=1}^{N}\gamma_k(n)\Bigg\{ & z_{n,i}\beta_{k,i}(0,1)\frac{-1}{1-\pi_{k,i}}+(1-z_{n,i})\frac{-1}{1-\pi_{k,i}} \\
& +z_{n,i}\beta_{k,i}(1,1)\frac{1}{\pi_{k,i}}\Bigg\}.
\end{aligned}
\tag{5.14}
$$

Setting Eq. (5.14) to zero and solving it, re-estimation formula for dropping rate $\pi_{k,i}$ is readily obtained

$$\frac{\partial}{\partial \pi_{k,i}} Q(\lambda; \lambda^{(\kappa)}) \bigg|_{\pi_{k,i}=(\pi_{k,i})^{(\kappa+1)}} = 0 \tag{5.15}$$

$$\Rightarrow (\pi_{k,i})^{(\kappa+1)} = \frac{\sum_{n=1}^{N} \gamma_k(n) z_{n,i} \beta_{k,i}(1,1)}{\sum_{n=1}^{N} \gamma_k(n)} \tag{5.16}$$

The derivative of the auxiliary function w.r.t. the i-th row $\mathbf{w}_i^{(c(k))}$ of $\mathbf{W}^{(c(k))}$ is computed as follows

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}_i^{(c(k))}} Q(\lambda; \lambda^{(\kappa)}) &= \sum_{n=1}^{N} \gamma_k(n) \left\{ \frac{z_{n,i} \beta_{k,i}(0,1)}{I_0(\lambda_{k,i}^{(\kappa)})} \int_{-\infty}^{c} \frac{y_{n,i} - \mathbf{w}_i^{(c(k))} \boldsymbol{\xi}_k}{\sigma_{k,i}^2} \boldsymbol{\xi}_k^T \mathcal{N}(y_{n,i}; \lambda_{k,i}^{(\kappa)}) \mathrm{d}y_{n,i} \right. \\
&\qquad \left. + (1 - z_{n,i}) \frac{x_{n,i} - \mathbf{w}_i^{(c(k))} \boldsymbol{\xi}_k}{\sigma_{k,i}^2} \boldsymbol{\xi}_k^T \right\} \\
&= \sum_{n=1}^{N} \gamma_k(n) \left\{ \frac{z_{n,i} \beta_{k,i}(0,1)}{\sigma_{k,i}^2 I_0(\lambda_{k,i}^{(\kappa)})} \left(I_1(\theta_{k,i}^{(\kappa)}) - I_0(\theta_{k,i}^{(\kappa)}) \mathbf{w}_i^{(c(k))} \boldsymbol{\xi}_k \right) \boldsymbol{\xi}_k^T \right\} \\
&\quad + \sum_{n=1}^{N} \gamma_k(n) \left\{ (1 - z_{n,i}) \frac{x_{n,i} - \mathbf{w}_i^{(c(k))} \boldsymbol{\xi}_k}{\sigma_{k,i}^2} \boldsymbol{\xi}_k^T \right\}.
\end{aligned} \tag{5.17}$$

The re-estimation formula for $\mathbf{w}_i^{(c(k))}$ is readily obtained by setting Eq. (5.17) to zero and solving it:

$$\begin{aligned}
&\sum_{n=1}^{N} \frac{\gamma_k(n)}{\sigma_{k,i}^2} \left(z_{n,i} \frac{I_1(\lambda_{k,i}^{(\kappa)})}{I_0(\lambda_{k,i}^{(\kappa)})} \beta_{k,i}(0,1) + (1 - z_{n,i}) x_{n,i} \right) \xi_k^T = \\
&\quad \sum_{n=1}^{N} \frac{\gamma_k(n)}{\sigma_{k,i}^2} \left(1 - z_{n,i} \beta_{k,i}(1,1)\right) \left(\mathbf{w}_i^{(c(k))} \right)^{(\kappa+1)} \xi_k \xi_k^T,
\end{aligned} \tag{5.18}$$

To estimate $\left(\mathbf{w}_i^{(c(k))} \right)^{(\kappa+1)}$ for a regression class $c(k)$-th which contains R positions $\{k_1, k_2, \cdots, k_R\}$, Eq. (5.18) becomes

$$\begin{aligned}
&\sum_{r=1}^{R} \sum_{n=1}^{N} \frac{\gamma_{k_r}(n)}{\sigma_{k_r,i}^2} \left(z_{n,i} \frac{I_1(\lambda_{k_r,i}^{(\kappa)})}{I_0(\lambda_{k_r,i}^{(\kappa)})} \beta_{k_r,i}(0,1) + (1 - z_{n,i}) x_{n,i} \right) \xi_{k_r}^T = \\
&\quad \left(\mathbf{w}_i^{(c(k))} \right)^{(\kappa+1)} \sum_{r=1}^{R} \sum_{n=1}^{N} \frac{\gamma_{k_r}(n)}{\sigma_{k_r,i}^2} \left(1 - z_{n,i} \beta_{k_r,i}(1,1)\right) \xi_{k_r} \xi_{k_r}^T,
\end{aligned} \tag{5.19}$$

Given the observed data and the initial values of the parameters to be estimated,

$\left(\mathbf{w}_i^{(c(k))}\right)^{(\kappa+1)}$ can be obtained as follows

$$\left(\mathbf{w}_i^{(c(k))}\right)^{(\kappa+1)} = \sum_{r=1}^{R}\sum_{n=1}^{N}\frac{\gamma_{k_r}(n)}{\sigma_{k_r,i}^2}\left(z_{n,i}\frac{I_1(\lambda_{k_r,i}^{(\kappa)})}{I_0(\lambda_{k_r,i}^{(\kappa)})}\beta_{k_r,i}(0,1)+(1-z_{n,i})x_{n,i}\right)\xi_{k_r}^T \left(\sum_{r=1}^{R}\sum_{n=1}^{N}\frac{\gamma_{k_r}(n)}{\sigma_{k_r,i}^2}\left(1-z_{n,i}\beta_{k_r,i}(1,1)\right)\xi_{k_r}\xi_{k_r}^T\right)^{-1} \tag{5.20}$$

5.2.2. Variance Adaptation

Variance adaptation is performed after mean adaptation. It is noted that we assume the RSSI measurements from different APs are independent.

Let $\mathbf{\Sigma}_k$ be the diagonal covariance matrix of the Gaussian random vector modelling the RSSI readings of the APs at position ℓ_k. The adapted variance can be calculated as follows [70]:

$$\hat{\mathbf{\Sigma}}_k = \mathbf{B}_k^T\mathbf{H}^{(c(k))}\mathbf{B}_k \tag{5.21}$$

where $\mathbf{H}^{(c(k))}$ is the transform to be estimated and $\mathbf{B}_k$ is the Choleski factor of $\mathbf{\Sigma}_k$. Since $\mathbf{\Sigma}_k$ are diagonal covariance matrices, the re-estimation formula Eq. (5.21) simplifies to $\hat{\sigma}_{k,i}^2 = h_i^{(c(k))}\sigma_{k,i}^2; i = 1,\ldots,N_{AP}$.

To estimate $\mathbf{H}^{(c(k))}$, we again employ the EM algorithm. The expected log-likelihood of the complete data, given the observed, has the same form as in mean adaptation. However, here $\lambda_{k,i} = \{\pi_{k,i}, h_i^{(c(k))}, \theta_{k,i}\}$ with $h_i^{(c(k))}$ is the i-th component of the main diagonal of $\mathbf{H}^{(c(k))}$.

Using all in the expected log-likelihood function Eq. (5.12), we arrive at:

$$\begin{aligned} Q(\lambda;\lambda^{(\kappa)}) = \sum_{k=1}^{K}\sum_{n=1}^{N}\gamma_k(n)\sum_{i=1}^{N_{AP}}\Bigg\{ & z_{n,i}\beta_{k,i}(0,1)\ln(1-\pi_{k,i}) \\ & + \frac{z_{n,i}\beta_{k,i}(0,1)}{I_0(\lambda_{k,i}^{(\kappa)})}\int_{-\infty}^{c}\ln\left(\frac{1}{\sqrt{2\pi h_i^{(c(k))}\sigma_{k,i}^2}}\exp\left(-\frac{(y_{n,i}-\hat{\mu}_{k,i})^2}{2h_i^{(c(k))}\sigma_{k,i}^2}\right)\right)\mathcal{N}(y_{n,i};\lambda_{k,i}^{(\kappa)})\mathrm{d}y_{n,i} \\ & + (1-z_{n,i})\ln(1-\pi_{k,i}) + (1-z_{n,i})\ln\left(\frac{1}{\sqrt{2\pi h_i^{(c(k))}\sigma_{k,i}^2}}\exp\left(-\frac{(x_{n,i}-\hat{\mu}_{k,i})^2}{2h_i^{(c(k))}\sigma_{k,i}^2}\right)\right) \\ & + z_{n,i}\beta_{k,i}(1,1)\ln(\pi_{k,i})\Bigg\} \end{aligned} \tag{5.22}$$

The re-estimation fomula for $h_i^{(c(k))}$ is readily obtained by computing the derivative of

Eq. (5.22) w.r.t. $h_i^{(c(k))}$ and setting it to zero:

$$\left(h_i^{(c(k))}\right)^{(\kappa+1)} = \frac{1}{\sigma_{k,i}^2 \sum_{n=1}^{N} \gamma_k(n)\,(1 - z_{n,i}\beta_{k,i}(1,1))} \sum_{n=1}^{N} \gamma_k(n) \left\{ (1 - z_{n,i})(x_{n,i} - \hat{\mu}_{k,i})^2 \beta_{k,i}(0,0) + z_{n,i} \left(\frac{I_2(\lambda_{k,i}^{(\kappa)})}{I_0(\lambda_{k,i}^{(\kappa)})} - 2\frac{I_1(\lambda_{k,i}^{(\kappa)})}{I_0(\lambda_{k,i}^{(\kappa)})} \hat{\mu}_{k,i} + \hat{\mu}_{k,i}^2 \right) \beta_{k,i}(0,1) \right\}. \tag{5.23}$$

Here $\hat{\mu}_{k,i}$ are the new updated means which were discussed in the previous subsection.

As can be seen in the derivation, beside estimating the adaptation matrices for means Eq. (5.18) and variances Eq. (5.23), the dropping rates of the adaptation data are simultaneously estimated. Eq. (5.18) and Eq. (5.23) show that not only the observable measurements, but also the unobservable ones contribute to the estimate of the adaptation matrices. If neither censoring nor dropping occurs, i.e., $\pi_{k,i} = 0$ and $z_{n,i} = 0$, Eq. (5.18) and Eq. (5.23) reduce to the formulas for mean and variance adaptation presented in [70].

5.3. Regression Classes

In the previous section, the method for aligning training model with adaptation data in the presence of clipped and dropped data has been presented for the general case of multiple transformation matrices.

If the relationship between two sets of training data and adaptation data follows only one linear rule as mentioned in [61], only one set of adaptation matrices, i.e., mean adaptation matrix and variance adaptation matrix, for all states is needed.

However, according to our data preliminary study, the relationship of RSSI data measured by any pair of devices does not follow only one linear rule. Two possible main factors which influence the relationship between two sets of data are signal strength range and signal frequency. This is obvious because of the fact that the sensitivities of radio signal sensors depend on the strength and the frequency of the measured radio signal. To relax the linearity assumption, a clustering approach has been applied to seperate the user positions in multiple regression classes that share the same linear relationship, however different from the other clusters.

However, a critical issue is how to define the regression classes, within which the parameters of all states, i.e., fingerprint positions, change in the same manner. Since no prior knowledge of which models should share the same transformation rule is available, a criterion for finding regression classes must be found. Various criteria could be used for clustering, for example, the positions with similar RSSI PDFs could share the same linear rule. The similarity of the PDFs can be measured by computing the likelihood of observing the training data collected at one position given the training models of the others. Using these criteria, means and variances of the training models are employed. However, from our observation, the states, i.e., positions, which transform in the same manner have similar RSSI mean values, irrespective of the actual identity of the APs. As a result, those positions should be assigned into the same cluster. We thus apply a k-means clustering on the mean vectors $\boldsymbol{\mu}_k$

of the probability density function of all location ℓ_k to obtain C clusters. For each cluster, transformation matrices are estimated as presented in the previous section.

6. Hidden Markov Model for Indoor User Tracking

6.1. Motivation

In chapter 4, we have discussed the method to estimate the parameters of censored and dropped Gaussian data and the optimal classification rule for localizing indoor users. The proposed algorithms belong to the most common approach of indoor positioning which is the fingerprinting based approach. Fingerprinting techniques are able to produce a stable positioning accuracy over time and movement distance of the user. However, there is another approach which can also be applied successfully in indoor positioning, namely the Dead Reckoning technique. This technique uses data from inertial sensors to locate the user position. However, as discussed in chapter 2, the Dead Reckoning technique is often applied in a combination with other positioning techniques since it can only produce precise position estimates in a short period of time and small distance. For long term use and large distance, the position error is unreasonably large because the errors are accumulated over time and distance.

As discussed, the stability of fingerprinting based indoor positioning techniques make them become the common candidates to accompany with DR techniques in indoor positioning. Various combinations have been proposed to keep the advantages and to reduce the disadvantages of those two approaches. Sensor fusion can be achieved by a Kalman filter [31], particle filter [32] or with the use of a Hidden Markov Model (HMM) [29, 30]. Among those possible solutions, we decided to employ the HMM in our system because it gives us the possibility to employ the knowledge of possible walking path which may improve the positioning accuracy.

6.2. Hidden Markov Model for Indoor User Tracking

In this section, the method to employ a HMM to fuse RSSI measurements and step detection information for position estimation is presented.

First, it should be recalled that the standard HMM can be depicted as in Fig. 6.1. Here, s_t is the hidden state variable and $\mathbf{x}_t$ is the observation at time t. It is noted that the hidden states cannot be observed, however, the most likely sequence of states can be inferred from the sequence of observations generated by HMM. The HMM is determined by three sets of parameters: the emission probabilities, the state transition probabilities and the initial state probabilities. Fig. 6.2 shows the unfolding of a state diagram over time, called trellis diagram. Here it is assumed that the HMM has $K = 4$ different states, i.e., $K = 4$ different values

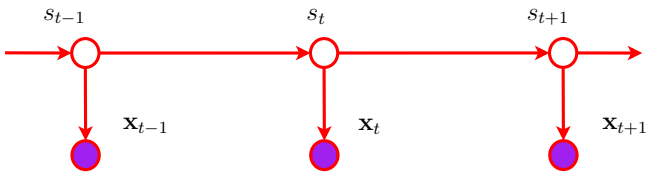


Figure 6.1.: Standard Hidden Markov Model: s_t is the hidden state variable and $\mathbf{x}_t$ is the observation at time t

of s_t: $s_t \in \{1, 2, 3, 4\}$. Each column in the graph (corresponding to a time instant) contains nodes representing the states of the HMM. Each node in the graph has connections to at least one node at an earlier and one node at a later time. The connections represent the possible transitions between states. The transitions from one state at one time instant to the possible states at the later time instant have probabilities which sum up to 1, i.e., $\sum_{j=1}^{K} P_{i,j} = 1$, where $P_{i,j} = P(s_{t+1}{=}j|s_t{=}i)$, assuming the transition probabilities are independent of time, given any $i = 1, \ldots, K$. Without any prior knowledge, the transition probabilities from one state to the next can be assumed to follow a uniform distribution.

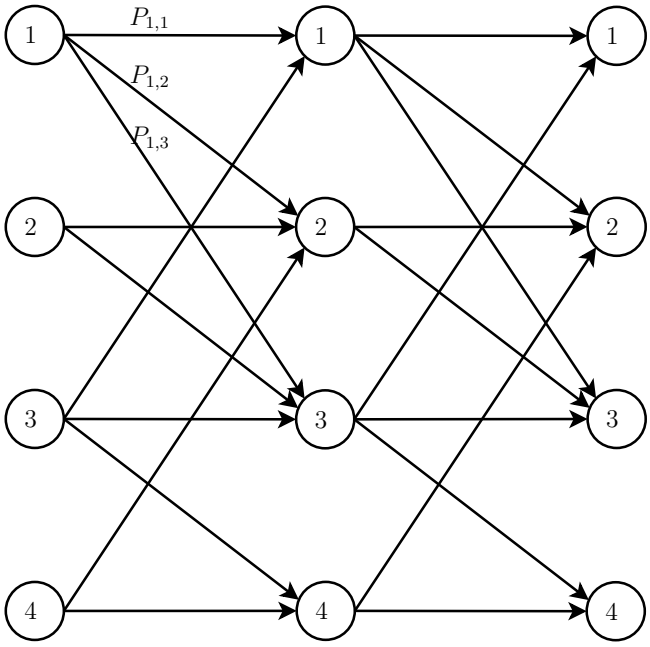


Figure 6.2.: Trellis diagram with four different states $s_t \in \{1, 2, 3, 4\}$ and $P_{i,j}$ are the transition probabilities from state i at one time instant to state j at the later time instant

The remaining parameters of a HMM are the emission probability distributions. For each HMM state, the emission probability distribution $p(\mathbf{x}|s_t)$ gives the likelihood of observing $\mathbf{x}$ at state s_t. Those emission PDFs are supposed to be different amongst states. For positioning purpose, we identify s_t with the position of the user at time t: if $s_t = k$ then the user is at the location ℓ_k at time t, where ℓ_k is a two-dimensional vector describing the user's location. In WiFi fingerprinting based indoor positioning system, the RSSI training models and the online RSSI measurements can be considered as emission probabilities and observation sequence, respectively.

However, with step detection information, there is another set of observations, i.e., move-

ment observations. To incorporate the step detection information, the emission probability distributions for this kind of observation need to be defined. Intuitively step detection information gives us the information about moving from one state to another state, i.e., the state transitions, instead of the information about a single state. So the solution here is to associate an observation with a state transition rather than a state. This ends up with a modification to the HMM to combine RSSI and step detection observations as depicted in Fig. 6.3. In the following, this model is called modified HMM.

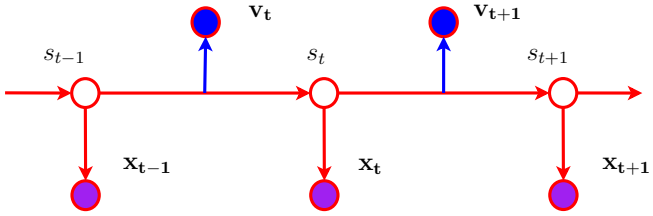


Figure 6.3.: Modified HMM for position estimation based on the fusion of RSSI and movement vector observations $\mathbf{x}$ and $\mathbf{v}$, respectively.

Here, $\mathbf{v}_t$ is the two-dimensional movement vector which denotes the traversed route from s_{t-1} to s_t. $\mathbf{v}_t$ is obtained from inertial sensor measurements. Its computation will be discussed in section 6.4. In the modified HMM, RSSI measurements and movement vectors are taken as observations attached to the hidden states and the state transitions, respectively. The transition probabilities between the states are chosen to reflect which positions are accessible from a given state within one measurement interval. Only transitions to those positions which are accessible within one measurement interval will be given a probability greater than zero. With the Samsung Android smartphones we have at hand, the measurement interval is roughly $1{,}5\,\mathrm{s}$ which is the required time to update a new WiFi scan of the devices.

6.3. Forward Algorithm

With the employment of a HMM, the estimation of the user position can then be carried out either by the Forward algorithm, the Viterbi algorithm or the Forward-Backward algorithm. While Forward algorithm computes the probability of being in a certain state by gathering the probabilities over all possible predecessor states, the Viterbi algorithm considers only the most probable predecessor. The Forward-Backward algorithm is only of academic interest because the induced latency is not acceptable for an online positioning system. In the following, the Forward algorithm which aims to estimate the most probable state at time instant t given the history of observations was chosen to decode the user position.

Let $\mathbf{x}_{1:t} = [\mathbf{x}_1, \ldots, \mathbf{x}_t]$ be the sequence of RSSI measurements up to time t. The step detection information is gathered in the sequence $\mathbf{v}_{1:t} = [\mathbf{v}_1, \ldots, \mathbf{v}_t]$. Our goal is to compute $P(s_t{=}k|\mathbf{v}_{1:t}, \mathbf{x}_{1:t})$, i.e., the probability of being in the k-th state for all possible user positions $\ell_k, k = 1, \ldots, K$, given all RSSI values and step detection vectors measured sofar. Using Bayes' rule, the probability can be expressed as follows:

$$\begin{aligned} P(s_t{=}k|\mathbf{v}_{1:t}, \mathbf{x}_{1:t}) &= \frac{p(s_t{=}k, \mathbf{v}_{1:t}, \mathbf{x}_{1:t})}{p(\mathbf{v}_{1:t}, \mathbf{x}_{1:t})} \\ &\propto p(s_t{=}k, \mathbf{v}_{1:t}, \mathbf{x}_{1:t}) =: \alpha_t(k), \end{aligned} \tag{6.1}$$

where the so-called Forward variable $\alpha_t(k)$ is the probability of being at time t in state k, while having observed the sequence of $\mathbf{x}_{1:t}$ and $\mathbf{v}_{1:t}$.

Using marginal probability and chain rule, the forward variable can be written as follows

$$\begin{aligned}\alpha_t(k) &= \sum_i p(s_t{=}k, s_{t-1}{=}i, \mathbf{v}_{1:t}, \mathbf{x}_{1:t}) \\ &= \sum_i p(\mathbf{v}_t|s_t{=}k, s_{t-1}{=}i, \mathbf{v}_{1:t-1}, \mathbf{x}_{1:t-1}, \mathbf{x}_t) \\ &\quad \cdot p(\mathbf{x}_t|s_t{=}k, s_{t-1}{=}i, \mathbf{v}_{1:t-1}, \mathbf{x}_{1:t-1}) \\ &\quad \cdot P(s_t{=}k|s_{t-1}{=}i, \mathbf{v}_{1:t-1}, \mathbf{x}_{1:t-1}) \\ &\quad \cdot p(s_{t-1}{=}i, \mathbf{v}_{1:t-1}, \mathbf{x}_{1:t-1}). \end{aligned} \tag{6.2}$$

Applying the properties of the HMM, which are depicted in the graphical model of Fig. 6.3, i.e., $\mathbf{x}_t$ is independent of all other variables if s_t is given, s_t is independent of $\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$ and $\mathbf{v}_1, \ldots, \mathbf{v}_{t-1}$ if s_{t-1} is given, and $\mathbf{v}_t$ is independent of everything if s_{t-1} and s_t are given, and assuming the step detection and RSSI information to be statistically independent of each other given the user location, the above formula simplifies to

$$\begin{aligned}\alpha_t(k) = &\sum_i p(\mathbf{v}_t|s_t{=}k, s_{t-1}{=}i) \cdot p(\mathbf{x}_t|s_t{=}k) \\ &\cdot P(s_t{=}k|s_{t-1}{=}i) \cdot \underbrace{p(s_{t-1}{=}i, \mathbf{v}_{1:t-1}, \mathbf{x}_{1:t-1})}_{=\alpha_{t-1}(i)} \end{aligned} \tag{6.3}$$

which is a recursion of the forward variable.

The final location estimate $\hat{\ell}$ is obtained by the weighted average over the set $\mathcal{P}$ of the most likely positions:

$$\hat{\ell} = \frac{1}{\sum\limits_{k\in\mathcal{P}} \alpha_t(k)} \sum_{k\in\mathcal{P}} \alpha_t(k) \cdot \ell_k. \tag{6.4}$$

Equation (6.3) shows how the different knowledge sources are combined. The transition probabilities $P(s_t{=}k|s_{t-1}{=}i)$ are nonzero only for those locations ℓ_k that can be reached from position ℓ_i within one time step. The choice of the transition probabilities thus encodes our knowledge about the floor plan. The term $p(\mathbf{x}_t|s_t{=}k)$ is the likelihood of the RSSI measurement $\mathbf{x}_t$, assuming that the user's position is ℓ_k. The computation of emission PDF estimation and likelihood calculation were discussed in Chapter 4. The movement information gathered from the step detection is captured by the term $p(\mathbf{v}_t|s_t{=}k, s_{t-1}{=}i)$, i.e., the likelihood of observing the movement vector $\mathbf{v}_t$ when moving from position ℓ_i to ℓ_k.

It is noted that this derivation requires that RSSI measurements and the movement information are obtained at the same rate. In fact, the rate of user's steps is often higher than RSSI measurements. To synchronize these 2 sources of data, movement vector $\mathbf{v}_t$ is the accumulated results of the step information during 1 RSSI measurement interval.

6.4. Movement Vector Estimation

In this section, the method to infer the step detection information from data measured by acceleration sensor, gyroscope sensor and magnetic sensor is summarized. More detailed information about movement vector estimation can be found in [71].

The system to estimate the movement vectors is depicted in Fig. 6.4.

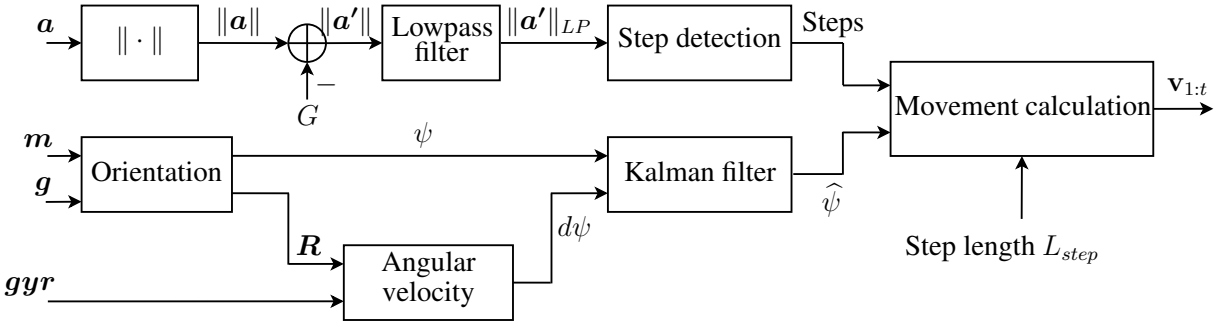
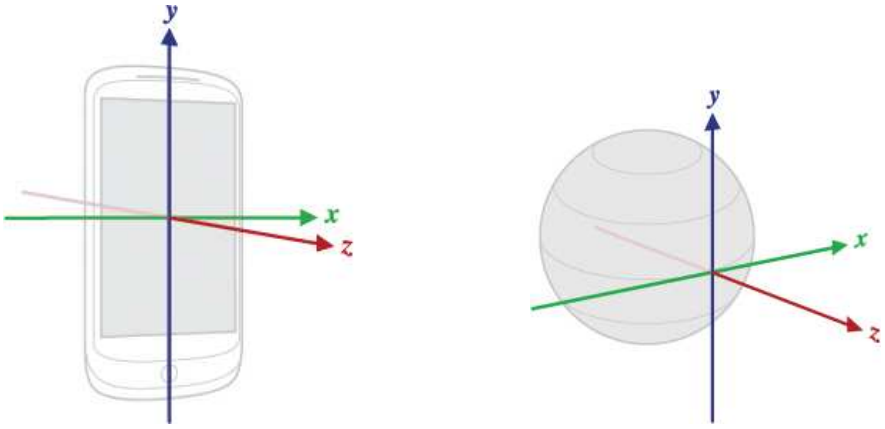


Figure 6.4.: Step detection and position estimation system overview

Here, $\boldsymbol{a}$ is the 3-dimentional acceleration vector from the accelerometer of the smartphone, $G = 9{,}81\,\mathrm{m/s^2}$ is the gravity constant, which will be used to perform the step detection procedure. $\boldsymbol{m}$, $\boldsymbol{g}$ and $\boldsymbol{gyr}$ are the magnetometer data, gravity information and gyroscope data, respectively, will be used to estimate the movement heading of the user. The gravity information vector $\boldsymbol{g}$ is the 3-dimentional vector which contains the force of gravity along three axes of the smartphone. $\boldsymbol{R}$ is the rotation matrix, ψ is the angle between movement heading and the magnetic north direction, and $d\psi$ denotes the changes of ψ over time. The detail information about these data and how to retrieve them by an Android application can be found on the website for android developers [72]. Movement vector $\boldsymbol{v}_t$ is computed by accumulating the estimated movement of all detected steps within one RSSI measurement interval.

6.4.1. Step Detection

The step detection procedure is based on the measured acceleration data from the smartphone. The obtained samples are stored in 3-dimensional vectors $\boldsymbol{a} = [a_x, a_y, a_z]^T$ where x, y, z are the axes in the coordinate system. This is defined in the relation with the screen of the smartphone as shown in Fig. 6.5(a)



(a) Device coordinates system (b) World coordinate system

Figure 6.5.: Device coordinate system (a) and world coordinate system (b)

The characteristics of the user movement is captured in the acceleration data as depicted in Fig. 6.6. As can be seen in the example the acceleration component along the z-axis clearly shows the up and down motion caused by walking. It is noted that the measured data are always influenced by the force of gravity. Therefore, to obtain the real acceleration data, the contribution of the force of gravity must be eliminated. One can use this a_z for step detection purpose if that is the stable feature of the obtained data. However, this is just an example of acceleration data when the smartphone was held in the way that its screen is in parallel with the earth surface. If the smartphone, however, is held in an arbitrary attitude, the observed data will not follow the example data in the figure any more. Because of that reason, instead of using data from one specific component, the Euclidean norm of the acceleration data from all three components is used to perform step detection, see Eq. (6.5). Fig. 6.7 shows the calculated norm values after substracting the force of gravity constant G=9.81.

$$\|\boldsymbol{a}\| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{6.5}$$

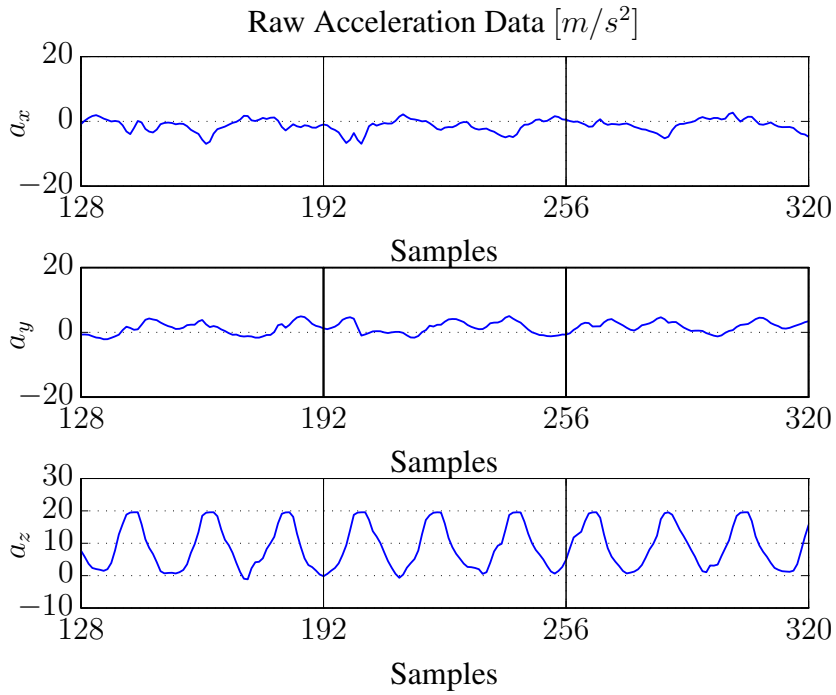


Figure 6.6.: Example of raw acceleration data if the smartphone is held with display in parallel to earth surface when walking

To reduce the variation caused by sensor errors, a 5-Hz lowpass filter is applied, as is suggested in [57]. Fig. 6.8 shows the output signal of the lowpass filter. Here, a Butterworth low-pass of order 20, with a cutoff frequency of $0.2 \cdot f_n = 5Hz$, where f_n corresponds to a half of the sampling rate $(50Hz/2)$ is used.

In the literature, two common approaches which have been employed in many research to detect steps using acceleration data are peak detection and zero crossing count. In [57] a peak detection with a minimum threshold is used to count the steps. This has the disadvantage that sometimes the maximum is split into two local maxima and thus an additional step is detected. The other method for detecting steps calculates the zero crossing rate of the

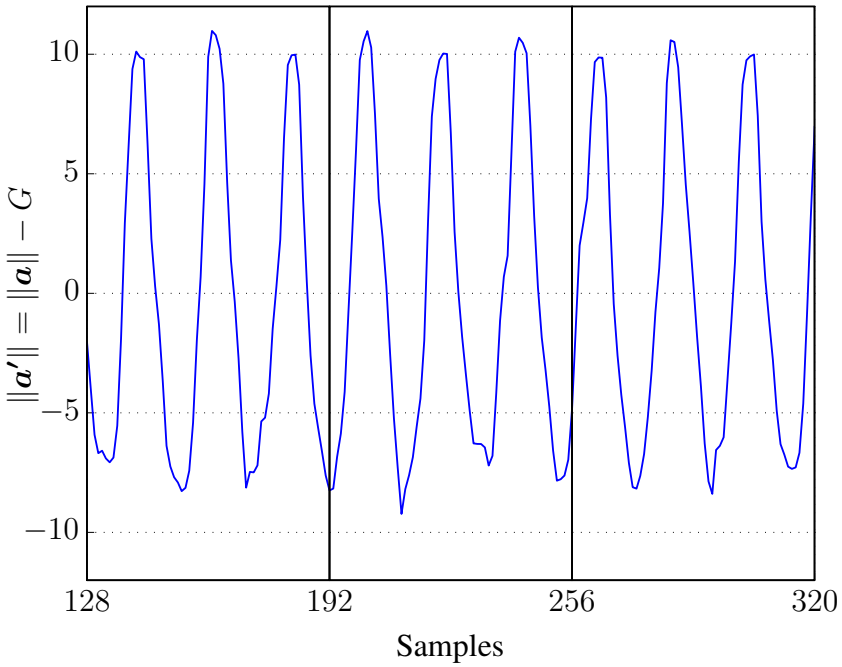


Figure 6.7.: Norm of acceleration data after substracting the force of gravity constant G=9.81

acceleration data [59], which in case of noisy data may also result in additionally detected steps.

The proposed approach combines the strengths of peak detection and zero crossing count approaches by counting the crossing rate of acceleration data over a certain threshold. Our method is then able to cope with multiple local maxima and noisy data in the measured acceleration data.

Fig. 6.9 illustrates the threshold-crossing approach. The step counter is only increased if the threshold (magenta line) is larger than the safety threshold (red line) to avoid an erroneous counting when the user is not moving but there is still some fluctuation of the observed acceleration data. The threshold is not a fixed value through the whole process, instead it is calculated for every single data block. Here a block size of 64 samples is chosen regarding the sampling rate of acceleration sensor and processing speed. First, for each data block, the peak mean is determined by calculating the mean of all the detected peaks. Second, the threshold is computed by multiplying the peak mean with an experimentally determined weighting factor. In our experiment, the weighting factor of 0.6 produces the best results.

Recently, another method, namely autocorrelation method, for detecting steps has been proposed in [32]. This method employs the fact that the acceleration data exhibits a very repetitive pattern. The advantage of the autocorrelation method compared to peak detection and zero crossing count methods is that it is able to eliminate the hand gestures when the user is not moving, transition from sitting to standing and vice versa, and so on. According to our experimental results, the autocorrelation method outperforms the other two traditional methods. Therefore, at the current state of our project, this method is employed in our system for detecting steps.

However, this section aims to present our proposed approach to detect steps which has be-

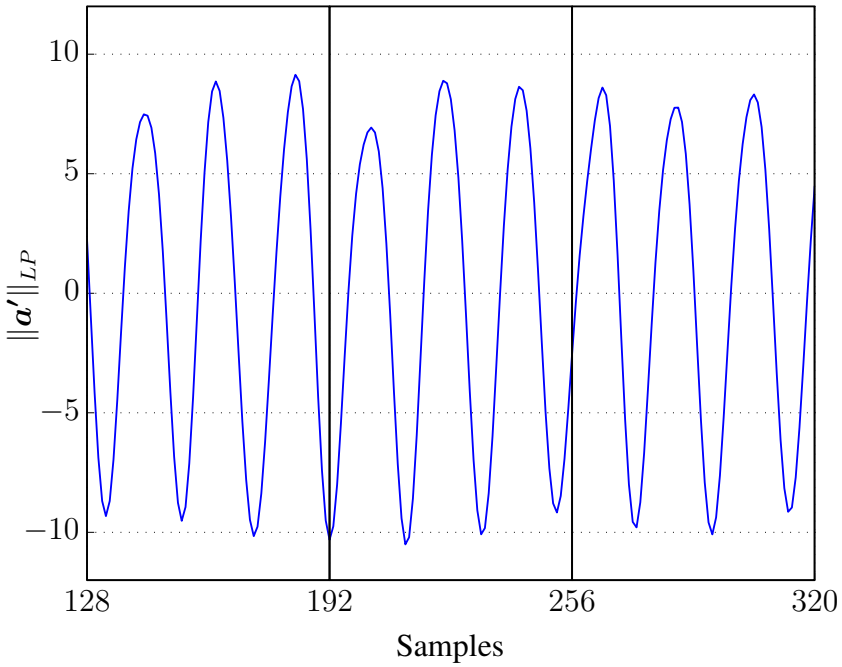


Figure 6.8.: $\|a'\|$ after lowpass filter

en reported in [73]. It has to be noted that the autocorrelation method has not been published when we started to develop our step detection method.

6.4.2. Movement Heading Estimation

Movement heading can be estimated by combining information from magnetic field sensor, gravity sensor and gyroscope sensor of the smartphone. Assuming that the user holds the smartphone with its screen in parallel with earth surface, i.e., the z-axis points toward the sky, perpendicular with the earth surface, and the y-axis points toward the movement direction of the user, the movement heading estimation simplifies to the estimation of the Azimuth angle ψ. This is the angle between the magnetic north direction and the y-axis (the smartphone rotates around its z-axis). Since the data from smartphone's sensors correspond to the device coordinate system, a rotation matrix R is needed to project observed data to the world coordinate system (see Fig. 6.5(b)). This rotation matrix can be obtained via the Android APIs using the measured gravity and magnetic data.

For simple movement heading estimation, one can use the estimated ψ from magnetic field and gravity information which gives the absolute angle between movement direction and the north pole. However, in indoor environment, movement heading estimation employing only magnetic information is very unstable due to the influence from many nearby metallic objects, i.e., elevators.

In order to obtain a more reliable and stable movement direction estimation, the combination of magnetic data and gyroscope data which are both available on most smartphones is chosen. Fusion of these two sets of data can be done by applying a Kalman filter.

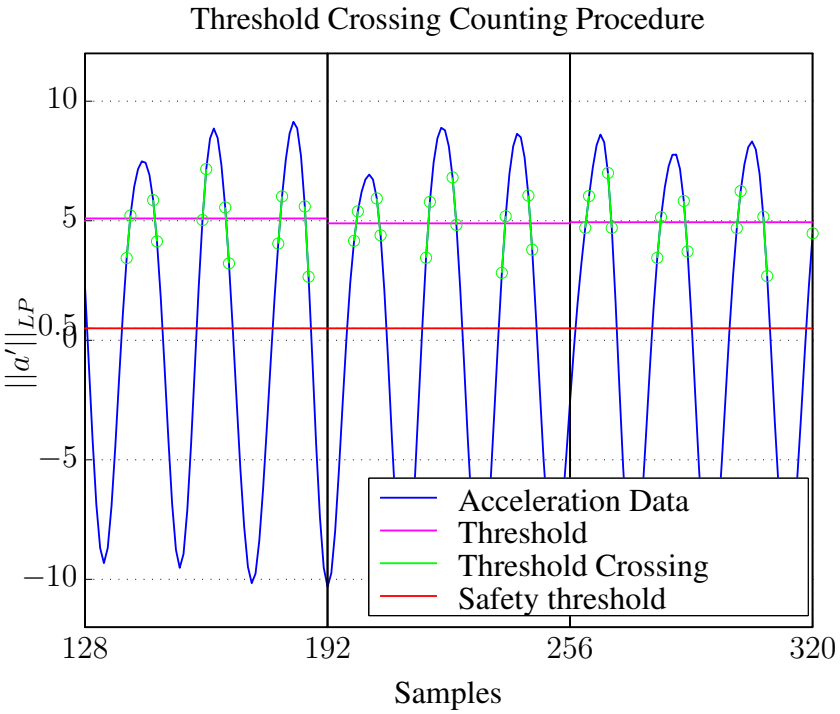


Figure 6.9.: Step detection procedure: In each data block, the number of steps is determined based on the crossing rate of the acceleration data over a threshold (magenta lines). The safety threshold is the minimum threshold value that the amplitude of noisy acceleration data cannot exceed.

The system equation of Kalman filter for sensor data fusion can be written as follows:

$$\boldsymbol{\alpha}(n+1) = \mathbf{F}(n)\boldsymbol{\alpha}(n) + \mathbf{G}(n)\boldsymbol{\nu}_1(n), \tag{6.6}$$

where $\boldsymbol{\alpha}(n)$ is the state vector that contains the absolute angle ψ and its devirative $\dot{\psi}$. $\mathbf{F}(n)$ is the state transition matrix which indicates a transition of the system from time n to $n+1$. $\boldsymbol{\nu}_1(n)$ is the system noise which is assumed to be white Gaussian noise that influences the state vector via the matrix $\mathbf{G}(n)$. The covariance matrix of $\boldsymbol{\nu}_1(n)$, $\mathbf{Q}_1$, is determined experimentally.

The measured sensor data $\boldsymbol{z}(n) = (\psi, \dot{\psi})^T$ are related to the state vector by the following measurement equation:

$$\boldsymbol{z}(n) = \mathbf{H}^H(n)\boldsymbol{\alpha}(n) + \boldsymbol{\nu}_2(n) \tag{6.7}$$

where $\mathbf{H}^H(n)$ is known as measurement matrix. Measurement noise is denoted by $\boldsymbol{\nu}_2(n)$ which is also assumed to be white Gaussian noise. The covariance matrix $\mathbf{Q}_2$ of $\boldsymbol{\nu}_2(n)$ is determined experimentally.

Within this work, the matrices $\mathbf{F}(n)$, $\mathbf{G}(n)$ and $\mathbf{H}^H(n)$ are assumed to be time invariant, and the values of those matrices, as well as $\mathbf{Q}_1$ and $\mathbf{Q}_2$, are provided in Appendix A.2.

6.4.3. Movement Vector Calculation

As mentioned previously, to be able to fuse inertial sensor information with RSSI measurements, those two sets of data need to be synchonized. Here we use RSSI scan interval T_s as

the common interval. The movement information of the user within time interval T_s can be computed as follows

$$\mathbf{v}_t = \sum_{t_i \in T_s} L_{step} \begin{pmatrix} sin(\psi(t_i)) \\ cos(\psi(t_i)) \end{pmatrix}, \tag{6.8}$$

where $t_i, i = 1, \ldots, N_{step}$ indicates the moments when steps are detected within T_s, N_{step} is the number of detected steps within T_s. L_{step} is the step length which has been estimated in advance, and $\mathbf{v}_t$ is the movement information of the user from time instant $t-1$ to t.

Obviously, using a fixed step length is not the optimal solution since the step length may vary according to different users or different walking styles of the same user. However, most step length estimation methods are based on the calibration procedures in the training phase which do not adapt well to different users either. Since the low cost built-in accelerometers on the smartphones have very limited accuracy, it seems that estimating step length accurately by double integration the acceleration data is unfeasible. Because of that, accurate on the fly step length estimation still needs to be carried out in the future work.

6.5. Introduction of Pseudo States

As mentioned in Chapter 4, the positioning accuracy using fingerprinting based method depends on how good the radio map is, i.e., how accurate the training parameters and how dense the positions with training data are. The EM algorithm was proposed to estimate the parameters of sensored and dropped data which is able to produce accurate parameter estimates, so the first issue is solved. The remaining issue is the density of the grid for which training data are gathered. Obviously, the more positions with training data, the better the performance of fingerprinting based positioning techniques [18, 25, 3]. However, it is very time consuming to develop an indoor positioning system for a large area, i.e., universities, hospitals, and so on, if we try to collect training data for a dense grid of positions. It is only feasible when we employ a fairly coarse grid of states, i.e., the average distance between two positions is about $3-5\,\mathrm{m}$.

The coarse grid of states leads to high quantization error. In order to obtain a good positioning accuracy without increasing training effort, we reduce the quantization error by introducing pseudo states in-between the regular HMM states of which the RSSI measurements have been collected during training. By doing this, the average distance between two possible user positions can be reduced. Fig. 6.10 illustrates our idea where states k and i are regular while the green one, i.e., state j, is the pseudo state. It is noted that the number of pseudo states inbetween two regular states depends on how far apart these two regular states are, and the designed maximum distance between two neighboring states after introducing pseudo states.

However, these pseudo states lack the emission PDFs since there are no training data. The PDFs of these pseudo states can be assumed to be related to their closest neighboring regular states. Assuming that the closeness in physical distance strongly influences the closeness in the measured signal strength, within this work a method was proposed to synthesize the emission PDF of a pseudo state by computing the product of the emission PDFs of the two closest neighboring regular states, where each PDF is raised to a power which has a value in

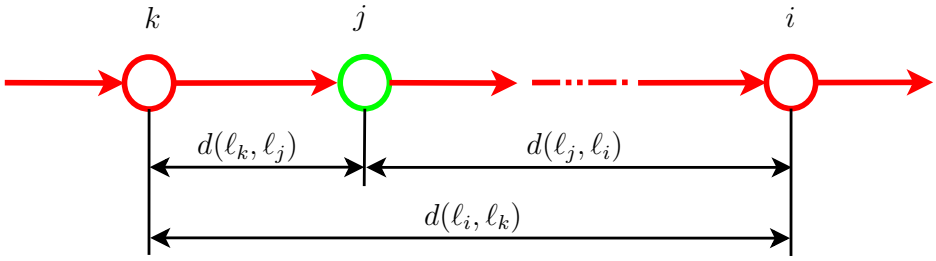


Figure 6.10.: Introduction of pseudo states: The red circles are the regular states with training data, the green circle is the pseudo state introduced inbetween the regular states.

the range from 0 to 1 and which depends on the distances of the pseudo state to the regular ones (see Eq. (6.9)).

$$p(x|s_t = j) = p(x|s_t = k)^{\frac{d(\ell_i,\ell_j)}{d(\ell_i,\ell_k)}} p(x|s_t = i)^{\frac{d(\ell_k,\ell_j)}{d(\ell_i,\ell_k)}} \tag{6.9}$$

Obviously, if the pseudo state is at the regular state, i.e., $j = i$, the emission PDF at the state j is the same as it at the state i. Eq. (6.9) intuitively gives the expected result $p(x|s_t = j) = p(x|s_t = i)$ since $d(\ell_i, \ell_j) = 0$ and $d(\ell_k, \ell_j) = d(\ell_i, \ell_k)$.

In case of $j \neq i, k$, the PDF at the j-th state can be computed from Eq. (6.9) as follows

$$\begin{aligned}
p(x|s_t = j) &= p(x|s_t = k)^{\frac{d(\ell_i,\ell_j)}{d(\ell_i,\ell_k)}} p(x|s_t = i)^{\frac{d(\ell_k,\ell_j)}{d(\ell_i,\ell_k)}} \\
&= \left(\frac{1}{\sqrt{2\pi\sigma_{\ell_k}^2}} \exp\left(-\frac{(x-\mu_{\ell_k})^2}{2\sigma_{\ell_k}^2} \right) \right)^{\frac{d(\ell_i,\ell_j)}{d(\ell_i,\ell_k)}} \left(\frac{1}{\sqrt{2\pi\sigma_{\ell_i}^2}} \exp\left(-\frac{(x-\mu_{\ell_i})^2}{2\sigma_{\ell_i}^2} \right) \right)^{\frac{d(\ell_k,\ell_j)}{d(\ell_i,\ell_k)}} \\
&= \left(\frac{1}{\sqrt{2\pi\sigma_{\ell_k}^2}} \right)^{\frac{d(\ell_i,\ell_j)}{d(\ell_i,\ell_k)}} \left(\frac{1}{\sqrt{2\pi\sigma_{\ell_i}^2}} \right)^{\frac{d(\ell_k,\ell_j)}{d(\ell_i,\ell_k)}} \\
&\quad \exp\left(-\frac{(x-\mu_{\ell_k})^2}{2\sigma_{\ell_k}^2} \frac{d(\ell_i,\ell_j)}{d(\ell_i,\ell_k)} - \frac{(x-\mu_{\ell_i})^2}{2\sigma_{\ell_i}^2} \frac{d(\ell_k,\ell_j)}{d(\ell_i,\ell_k)} \right)
\end{aligned} \tag{6.10}$$

Eq. (6.10) can be written in a better form as follows

$$\begin{aligned}
p(x|s_t = j) = &\left(\frac{1}{\sqrt{2\pi\sigma_{\ell_k}^2}} \right)^{\frac{d(\ell_i,\ell_j)}{d(\ell_i,\ell_k)}} \left(\frac{1}{\sqrt{2\pi\sigma_{\ell_i}^2}} \right)^{\frac{d(\ell_k,\ell_j)}{d(\ell_i,\ell_k)}} \\
&\exp\left(-\frac{1}{\frac{2\sigma_{\ell_k}^2 \sigma_{\ell_i}^2 d(\ell_i,\ell_k)}{d(\ell_i,\ell_j)\sigma_{\ell_i}^2 + d(\ell_k,\ell_j)\sigma_{\ell_k}^2}} \left(x - \frac{\mu_{\ell_k} d(\ell_i,\ell_j)\sigma_{\ell_i}^2 + \mu_{\ell_i} d(\ell_k,\ell_j)\sigma_{\ell_k}^2}{d(\ell_i,\ell_j)\sigma_{\ell_i}^2 + d(\ell_k,\ell_j)\sigma_{\ell_k}^2} \right)^2 \right) \\
&\exp\left(-\frac{(\mu_{\ell_k} - \mu_{\ell_i})^2 d(\ell_i,\ell_j) d(\ell_k,\ell_j)}{2d(\ell_i,\ell_k)\left(d(\ell_i,\ell_j)\sigma_{\ell_i}^2 + d(\ell_k,\ell_j)\sigma_{\ell_k}^2 \right)} \right) \\
&\propto \mathcal{N}(x|\mu_{\ell_j}, \sigma_{\ell_j}^2),
\end{aligned} \tag{6.11}$$

where

$$\frac{\mu_{\ell_j} d(\ell_i, \ell_k)}{\sigma^2_{\ell_j}} = \frac{\mu_{\ell_k} d(\ell_j, \ell_i)}{\sigma^2_{\ell_k}} + \frac{\mu_{\ell_i} d(\ell_k, \ell_j)}{\sigma^2_{\ell_i}} \tag{6.12}$$

$$\frac{d(\ell_i, \ell_k)}{\sigma^2_{\ell_j}} = \frac{d(\ell_j, \ell_i)}{\sigma^2_{\ell_k}} + \frac{d(\ell_k, \ell_j)}{\sigma^2_{\ell_i}}. \tag{6.13}$$

As the result obtained above, the synchonized PDF at the state j is again a Gaussian. The effectiveness of introducing pseudo states to the HMM on positioning accuracy is presented in chapter 7.

7. Experimental Results on Indoor Positioning

This chapter presents the experimental results on both artificially generated data and real field data to demonstrate the effectiveness of the proposed algorithms. The results presented in this chapter have been reported in some publications [74, 73, 75, 76]. At each location training data has been gathered by a user holding a smartphone. The user turned by 90 degree, such that measurements were taken at four different orientations. This was done to reflect the influence of the human body, antenna orientations, etc., in the training data. Since we have collected several sets of field data for different experiments, more detailed information about the data collection procedure of each set of field data will be given in each section.

7.1. Parameter Estimation and Classification

First, the effectiveness of the proposed EM algorithm for censored data has been evaluated. Comparison between the proposed methods and some other methods demonstrated that the proposed methods outperform the others (section 7.1.1). Secondly, the experimental results, which prove that being aware of dropping in addition to censoring improves the classification performance, are presented in section 7.1.2. Note that, in this section, training data and test data are collected by the same device, so the mismatch problem between training and test data arising from using different devices does not occur.

7.1.1. EM algorithm for censored data

Artificial Data

In the following the effectiveness of the EM algorithm for an indoor localization problem is evaluated using artificially generated data: We consider a 2-class problem with $N_{AP} = 5$ access points. For each of the two locations $N = 1000$ training samples are drawn from a normal density with parameters according to Table 7.1 and then censored from the left with a threshold of $c = -100$ (dBm). A total of 200 test samples, 100 per location, are generated in the same manner. Performance is measured in terms of classification error rate. For the generated data, the percentage of uncensored data, averaged over all APs at each position, was approximately 58%. A classification error occurs if the algorithm classifies the measurement to a different location than the location whose PDF of RSSI values it was drawn from.

We compared the classification error rate of the following schemes:

Table 7.1.: Mean and standard deviation of APs at 2 positions for artifical data experiment

AP index	AP1	AP2	AP3	AP4	AP5
$\mu_{1,i}$	-102	-103	-97	-89	-95
$\mu_{2,i}$	-105	-100	-99	-86	-101
$\sigma_{1,i}$	4.8	4.9	5.0	5.2	5.1
$\sigma_{2,i}$	5.0	4.8	4.8	5.4	5.0

Table 7.2.: Classification error rate on artificial censored data

Method	Error rate (%)
Plain trng + recog	30.7
EM trng + plain recog	26.9
EM trng + censored recog	**22.5**
3-strongest APs	35.1
1-nearest neighbor	36.8

- Plain training (trng) + recognition (recog): ML parameter estimation is carried out assuming normally distributed, uncensored data. Also recognition is performed disregarding any censoring.
- EM trng + plain recog: ML parameter estimation in training accounts for the censored data using the proposed EM algorithm, while the presence of censored data is still disregarded in recognition.
- EM trng + censored recog: Training with the proposed EM algorithm and recognition employing eq. (4.52).
- 3-strongest APs: Select three strongest APs of each location in the training phase, then apply EM trng + censored recog.
- 1-nearest neighbor classification rule.

Table 7.2 clearly shows the superiority of the schemes which are aware of the censoring. Considering the presence of censored data in training improved the error rate from 30.7% to 26.9%, and a further improvement to 22.5% is obtained by accounting for censored data also in recognition. We can also see the important role of weak APs for the recognition accuracy: using only the three strongest APs raises the error rate to 35.1%. 1-nearest neighbor performs worst with the error rate of 36.8%.

Field Data

In order to evaluate the performance of the proposed EM algorithm on real world data, real WiFi RSSI measurements were gathered on a floor of an office building consisting of 10 office rooms and a long aisle having an overall size of $12\,\mathrm{m}$ by $30\,\mathrm{m}$ (see Fig. 7.1). RSSI values were taken at 25 different positions, roughly evenly distributed, resulting in an average distance of $2{,}7\,\mathrm{m}$ between two locations. Two measurement campaigns were carried out using a smartphone, with 100 measurements taken per position per campaign. Data of the

first measurement campaign served the purpose of training and the second one was for testing purposes. For the training data set, the percentage of uncensored observations, averaged over all APs which were observable at each location, was found to be 36.7%.

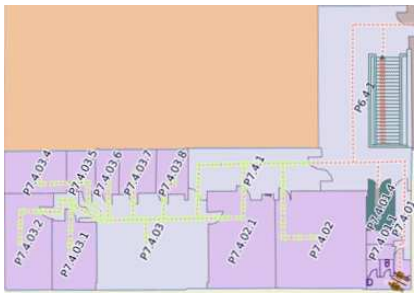


Figure 7.1.: Floor plan of the area where field data has been conducted.

To compare the proposed approach to a state-of-the-art system, the algorithm from [52] was implemented. There, for each location ℓ_k the probability distributions of the 10 strongest APs are determined during the training phase and compared to those of the online measurements employing the Bhattacharyya coefficient. A 3-nearest neighbor rule is then applied to decide on the user location. Furthermore, we compared with the well-known system, RADAR [18], where classification is performed with a 3-nearest neighbor rule, employing the Euclidian distance. When applying the proposed algorithm, 5 online measurements were used for each estimate, the final location estimate was then computed using the 3 most likely positions, see Eq. (4.53). It is noted that 5 online measurements were also employed in the implementation of the algorithm of [52].

Fig. 7.2 shows the cumulative distribution function (CDF) of the error as a function of the distance for each method. The CDF is defined as the probability that the positioning error ϵ is lower than a certain distance d:

$$\texttt{CDF}_\epsilon(d) = P(\epsilon \leq d) \quad d \geq 0. \tag{7.1}$$

The results in Fig. 7.2 show that the proposed method outperforms the other two, especially for the 40% error quantile. Note, also, that the computational cost of the proposed method during the online phase is smaller than those of computing the Bhattacharyya distances between probability distributions [52] or the nearest-neighbor based [18] methods.

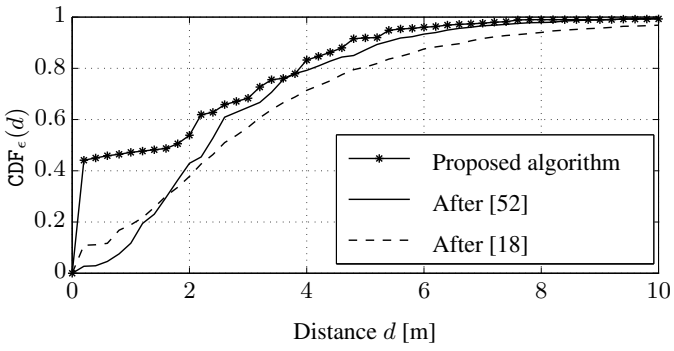


Figure 7.2.: CDF of the positioning error for different systems.

Table 7.3.: Classification error rate on artificial censored and dropped data

Method	Error rate (%)
EM trng + censored recog	29.7
Adv. EM trng + censored & dropped recog	**25.7**

7.1.2. EM algorithm for censored and dropped data

In the following, experimental results showing the effectiveness of EM algorithm when being aware of dropped data in addition to censored data are presented. For convenience, let us call the EM algorithm for parameter estimation of censored and dropped data as the advanced EM algorithm.

Artificial Data

The artificial data were generated according to the parameters which were used in Section 7.1.1, however, the means of all APs at all positions are increased by $10\,\mathrm{dBm}$ to show more impact of dropped data on the estimated parameters and consequently on classification results. The generated Gaussian data were first censored and then dropped with the dropping rate of 20%. Since the effectiveness of EM algorithm for censored data has been proved in the previous sections, this section aims to compare the classification results between two proposed EM algorithms to show the importance of the awareness of dropped data besides censored data.

Table 7.3 shows the classification results on censored and dropped data using the proposed EM algorithms. It can be seen that the parameter estimation formulas derived in Section 4.2.3 are able to cope with an unknown drop-out rate, which leads to the improvement in the classification error rate from 29.7% to 25.7% if indeed drop-outs occur. The reason is if all unobservable data are considered as censored data, the parameters estimated by the EM algorithm which is aware of censored data only, are inaccurate, i.e., the estimated means are biased to the left of the true means and the estimated variances are higher than the true variances. These inaccurate estimated parameters cause the degradation of the classification performance.

Field Data

To demonstrate the effectiveness of the advanced EM algorithm, an experiment on real field data has been done. This experiment was done by using the same data sets as used in the experiment of the EM algorithm for censored Gaussian data. Fig. 7.3 shows the experimental results on real field data using the two proposed EM algorithms. As can be seen, being aware of dropped data improves the positioning accuracy, though the improvement is moderate. The reasons might be either the amount of training data, 100 samples per position, are not sufficient for the advanced EM algorithm since it tries to estimate more parameters, or the true dropping rates are low. Since the data sets were gathered in a real indoor environment, true dropping rates are unknown. As our study on the results, the estimated means and variances from advanced EM are better than the EM algorithm awaring of censored data only, and the estimated dropping rate averaged over all APs and all locations was approximately

0.37, we concluded that the limited improvement of positioning accuracy is because of the not very accurate estimated dropping rates. As presented in section 4.3, dropping rate plays an important role in likelihood calculation for the unobservable data, so the inaccurate estimated dropping rates limit the improvement in classification results. In the bad case of low accuracy of the estimated dropping rates, the positioning accuracy might even decrease.

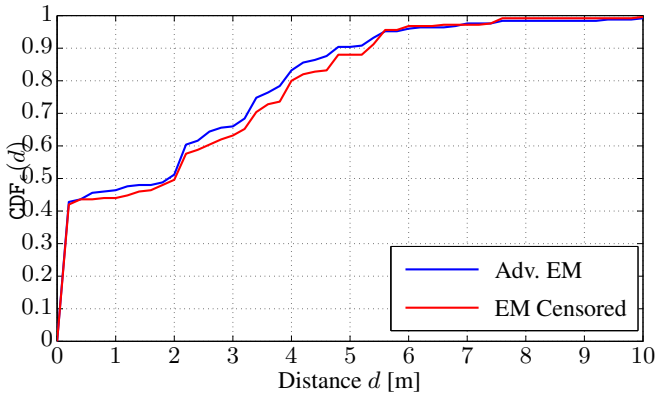


Figure 7.3.: Comparing the experimental results on real data using 2 proposed EM algorithms

7.2. Smartphone Adaptation

This section investigates the impact of the proposed adaptation approach on the accuracy of fingerprinting based indoor positioning. Some variations of the adaptation approach within the MLLR framework such as mean and variance adaptation, mean adaptation only with full or diagonal adaptation matrix have been implemented. In addition, the impact of the number of adaptation data and of the number of regression classes on the performance of adaptation, and consequently on the positioning accuracy, are also investigated.

7.2.1. Classification on Artificial Data

In this set of experiments, artificial data were generated to assess the impact of the drop-out rate and the amount of adaptation data on the classification performance.

1000 samples of training data were generated for each of $K = 6$ positions and for $N_{AP} = 2$ access points. A single affine transformation of the means of the training data is used for each location according to $\hat{\boldsymbol{\mu}}_k = \mathbf{A}\boldsymbol{\mu}_k + \mathbf{b}$, where $\mathbf{A} = [0.9, 0; 0, 0.9]$ and $\mathbf{b} = [-3, 2]^T$. Adaptation and test data were then generated by sampling from the transformed Gaussians with means $\hat{\boldsymbol{\mu}}_k$ and variances as those of the training data. While the amount of adaptation data was gradually increased from 1% to 5, 10, 50 and 100% of the number of training samples, the number of test samples was fixed at 100 observations per position. The value of the drop-out rate is set to 0% (no drop-outs, $\pi = 0$) and 20% ($\pi = 0.2$).

Although the effectiveness of being aware of dropped data on classification results was demonstrated in section 7.1.2, for the convenience of evaluating the effectiveness of the adaptation approach, classification results employing advanced EM with the current set of parameters are still presented. Table 7.4 discusses the impact of the drop-out rate without

Table 7.4.: Classification results (% positions correctly classified) when training and test data are generated from the same parameter set

Aware of dropping	yes	no
$\pi = 0$	86	86
$\pi = 0.2$	75	64

Table 7.5.: Adaptation performance (% positions correctly classified) on artifical data as a function of amount of adaptation data

No. Pos with adapt. data	π	0%	1%	5%	10%	50%	100%
3	0	68	81	85	85	86	86
	0.2	56	70	73	74	75	75
6	0	68	85	86	86	86	86
	0.2	56	72	75	75	75	75

adaptation. Here, the 'adaptation' data were used for a completely new training from scratch for each of the 6 positions. 1000 samples of adaptation data were used in this experiment. As can be seen, if no dropping occurs, the two EM algorithms produce the same classification result. Once the dropping occurs, the advanced EM algorithm outperforms the other.

Table 7.5 shows the effectiveness of applying the proposed adaptation algorithm. As can be seen, the results when only 3 locations have adaptation data is comparable to the case when adaptation data are available for all 6 locations. The reason is even if there are only 3 locations with adaptation data, still the PDFs of all locations are well adapted, since they are from a single regression class. However, it is also noticeable that when the number of adaptation data per position is small, i.e., less than 10% of the training data, classification results using 6 positions with adaptation data are about 1 to 4% better than those when adaptation data are only available for 3 positions. The column with 0% of adaptation data shows the results when no adaptation is performed. Comparing with the results of Table 7.4 it is clear that the classification results after adaptation approach those of retraining, except when there is only 1% of adaptation data, even if adaptation data are only available for 3 out of 6 positions. For the case of only 1% of adaptation data available, there is still a dramatic improvement in classification results when adaptation is employed compared to the case if no adaptation is performed.

7.2.2. Classification on Field Data

To examine the effectiveness of the adaptation approach on real world data, measurements were collected on 3 floors of an office building with roughly 30 rooms (lecture halls, office and laboratory rooms), where each floor has an overall size of $35\,\mathrm{m}$ by $35\,\mathrm{m}$, RSSI values were taken at 60 different positions with an average distance of 5.0 m between 2 positions. 200 measurements were taken per position with 2 different smartphones at each position. Data of the first smartphone were used to estimate the training models while data from the second smartphone were divided into 2 sets at each position. The first was used for adaptation (0, 5, 25, or 75 samples) or retraining (150 samples) from scratch and the second was for testing (50 samples). For the measured data, the estimated dropping rate averaged over all

Table 7.6.: RMS positioning error (in [m]) as a function of the amount of positions having adaptation data and the amount of adatation data

Condition		Adaptation Method		
No. Pos with adapt. data	Amount of adapt. data	μ & σ^2, **A** full	μ only, **A** full	μ only, **A** diag
	0	6.15		
15	5	5.08	4.76	3.93
	25	4.60	4.41	3.93
	75	4.62	4.52	3.93
30	5	3.86	3.96	3.84
	25	3.82	3.96	3.85
	75	3.76	3.91	3.85
60	5	3.74	3.93	3.84
	25	3.67	3.87	3.83
	75	3.65	3.87	3.84
	retrain	2.43		

APs and all positions was approximately 0.3.

For the adaptation procedure, the estimated training means were sorted at each position in descending order, and only the 8 strongest APs were used to estimate the adaptation matrices since the contribution of the remaining APs to the likelihood was negligible. The adaptation matrices are then used to calculate the adapted parameters of the 8 strongest APs at all positions being in the same regression class.

Table 7.6 shows the dependency of the positioning accuracy on the number of locations for which adaptation data are available and the amount of available adaptation data at each position. The results in Table 7.6 are the average of the root mean square (RMS) positioning error of 50 experiments. In each experiment, test data, adaptation data and the positions with adaptation data are randomly selected. As expected, the more positions there are with adaptation data and the more adaptation samples per position, the better the positioning results.

For all considered adaptation methods, improvements in positioning accuracy are obtained, even when very few adaptation data are available. However, mean and variance adaptation with a completely filled matrix $\mathbf{A}$ delivers only the best results if there are sufficiently many adaptation data, while the use of only mean adaptation with a diagonal $\mathbf{A}$ is superior if only few adaptation data are available, since fewer parameters need to be estimated. From the presented results, indoor positioning system developers could have an idea of which and how parameters can be efficiently adapted given the available adaptation data.

However, the positioning accuracy when all 60 positions have adaptation data is still well below the accuracy achievable, when training and test data are collected by the same smartphone, which is 2.43 m. The reason could be one transformation rule could not describe well the relationship between training and adaptation data for all positions which actually follows a nonlinear rule as discussed in chapter 5.

To relax the linearity assumption between training and adaptation data, multiple regression classes are employed. For estimating the regression classes, the means of the 8 strongest APs at each position are sorted in descending order, the resulting 8-dimensional vectors are

Table 7.7.: Effect of number of clusters on RMS positioning error

No. of clusters	1	2	3
RMS pos. error [m]	3.76	3.48	3.25

clustered using k-means, as discussed in section 5.3. Table 7.7 shows the positioning results when doing clustering and estimating different adaptation matrices for each cluster, assuming that in each cluster 50% of positions have adaptation data with 75 adaptation samples per position. It has to be noted that the optimal number of regression classes depends on the amount of available adaptation data. The more adaptation data the more transformation matrices can be reliably estimated. In our setup the best results were achieved with 3 clusters, which led to a reduction of the RMS positioning error from 3.76 m to 3.25 m.

7.3. HMM for Indoor User Tracking

This section evaluates the effectiveness of the proposed algorithms discussed in Chapter 6 for an indoor positioning problem both on artificially generated data and real field data, especially the impact of the introduction of pseudo states on the classification/positioning results.

7.3.1. Artificial Data

Fig. 7.4 shows the HMM states with the locations of the regular and pseudo states marked with red and green circles, respectively.

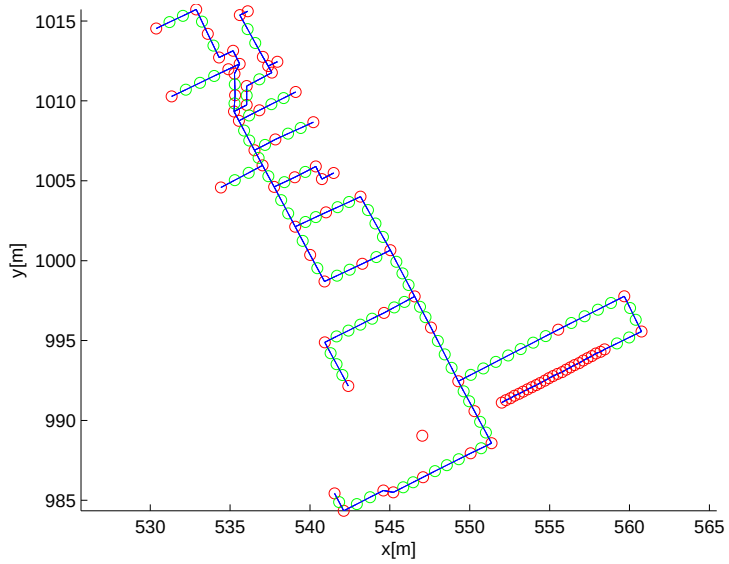


Figure 7.4.: The modified HMM states, i.e., the allowable user positions, and the transitions between them. Red and green circles indicate regular and pseudo states, respectively.

The RSSI measurements of 15 randomly placed APs for training the Gaussian emission PDF for the regular HMM states are generated artificially as follows: The signal strength fol-

Table 7.8.: Mean positioning error on artificial data

Method	Mean error [m]
RSSI only [74]	1.74
RSSI + step det. [73]	1.37
RSSI + step det. + pseudo states [75]	**1.02**

lows a large-scale log-normal fading model with an additional zero mean Gaussian random variable with standard deviation $\sigma_L = 5$ to model small-scale fading. At each position, we generated a set of 300 RSSI measurements as the training data, then estimated the parameters of RSSI distribution of each AP at each position as described in Chapter 4.

The step detection information is modeled as a bivariate Gaussian with the mean $\boldsymbol{\mu}_{i,j} = \ell_j - \ell_i$ and a diagonal covariance matrix $\boldsymbol{\Sigma}_v$ with entries $\boldsymbol{\Sigma}_v[1,1] = \boldsymbol{\Sigma}_v[2,2] = 0{,}25\,\mathrm{m}^2$. This value has been determined in offline experiments.

In the experiment, pseudo states were introduced in the way that the Euclidean distance between neighboring states is not more than at most $0{,}75\,\mathrm{m}$ (which is close to the measured average step length within the experimental data), while the distance between two neighboring regular states is about 3-5 m except for some special areas such as the stairs. The total number of pseudo states is 125, which has to be compared to the number of 81 regular states.

In the experiments we assume that a user cannot move faster than $3\,\mathrm{m/s}$. User movement is simulated by a random walk on the HMM graph. Since movement vectors and RSSI measurements are generated every $1{,}5\,\mathrm{s}$, only a limited number U of HMM states can be reached from any given state $s_{t-1}{=}i$ (on the average $U{=}15$ states) and the corresponding transition probability $P(s_t{=}j|s_{t-1}{=}i)$ is set to $\frac{1}{U}$. For all states outside this neighborhood the corresponding transition probabilities are set to zero.

Table 7.8 presents the mean positioning error in meters, averaged over 100 experiments, where each experiment corresponds to a different random walk on the HMM grid of length of about $200\,\mathrm{m}$. We compare the performance of the proposed algorithm with our earlier work of [73], which also fused RSSI and step detection information, however without the introduction of pseudo states. Using pseudo states improves the mean positioning error from $1{,}37\,\mathrm{m}$ to $1{,}02\,\mathrm{m}$. If step detection information is neglected and user positioning relies only on RSSI information, a mean positioning error of $1{,}74\,\mathrm{m}$ is obtained.

7.3.2. Field Data

The proposed approach is also evaluated with the field data recorded in the office building depicted in Fig. 7.1. In the training phase, 100 RSSI measurements per position were collected and the RSSI distribution were estimated as described in Chapter 4. In the online testing phase, the smartphone user randomly went through the whole floor area to collect the test data, i.e., WiFi data and inertial sensor data. Two different trajectories were recorded, each trajectory consists of approximately 140 test potitions. The position estimate was performed every $1{,}5\,\mathrm{s}$ using the proposed approach. The step detection information is modeled in the same fashion as in the experiments using artificial data.

According to the obtained data and experimental results, it seemed that the step detection information is more reliable than the RSSI information, as a consequence, a heuristic

weighting factor λ was introduced in the calculation of the forward variable as follows

$$\alpha_t(j) = [p(\mathbf{o}_t|s_t{=}j)]^{\lambda} \sum_i [p(\mathbf{v}_t|s_t{=}j, s_{t-1}{=}i)]^{(1-\lambda)} P(s_t{=}j|s_{t-1}{=}i) \cdot \alpha_{t-1}(i). \qquad (7.2)$$

For the determination of λ a jackknife procedure was employed: The data of 1 out of the 2 trajectories was used for the estimation of λ, whereas tests were conducted on the held-out data. This was repeated 2 times, every time, one trajectory was used to estimate λ. With our collected data, the estimated value was always $\lambda \approx 0.003$. The very small value of λ can be explained as follows: since the likelihood of RSSI data is much smaller than the likelihood of step detection information, this value of λ would help to avoid the problem that the contribution of the likelihood of step detection information to the calculated forward variable is dominated by the likelihood of RSSI data.

The proposed method was compared to our ealier work: first, using RSSI only for user positioning [74], and second, using HMM model as described in Chapter 6, however without the introduction of pseudo states [73]. For both trajectories the experimental results showed that the new approach outperforms the others, especially for the 90% error quantile, in terms of the CDF of the positioning error Eq. (7.1).

Although the test area is limited, the experimental results in Fig. 7.5 indicate that the proposed approach is significantly better than the other approaches.

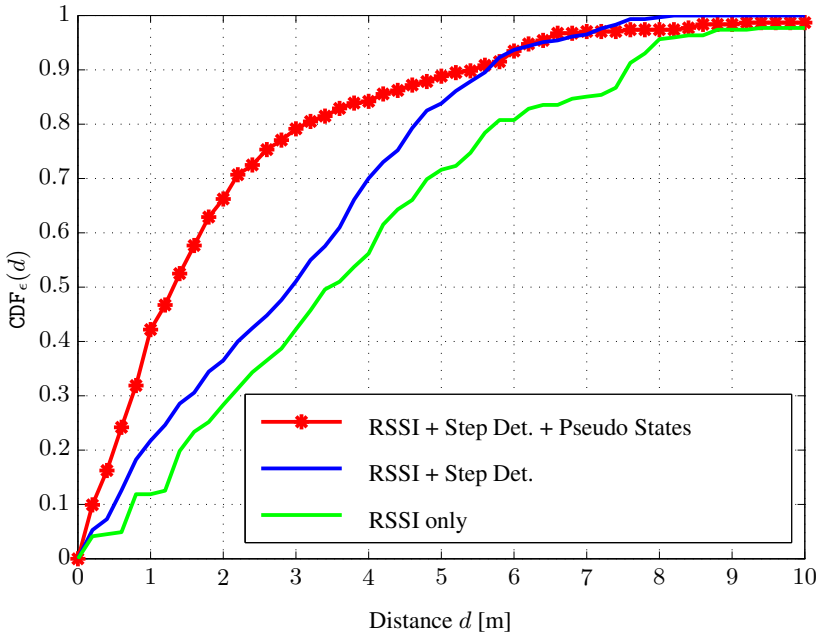


Figure 7.5.: CDF of the positioning error for different systems. Average over 2 test trajectories.

8. Server Based Indoor Navigation System

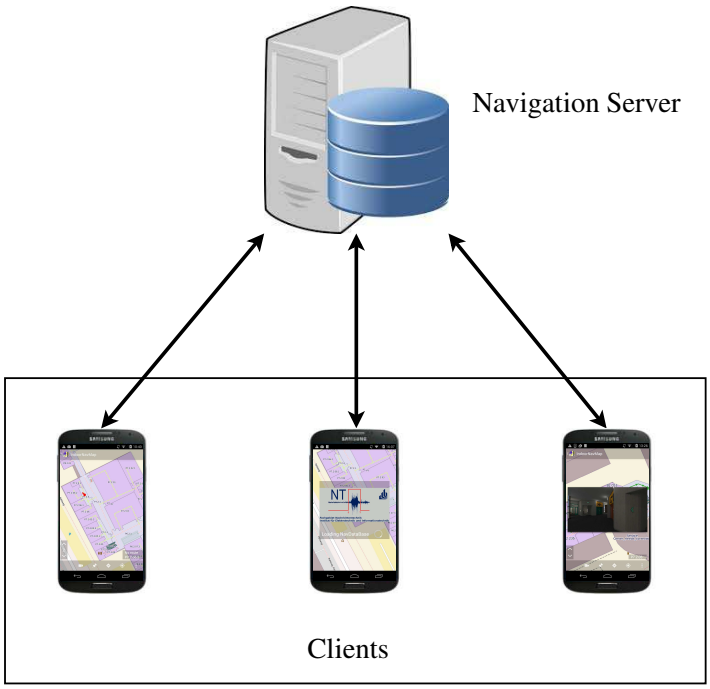


Figure 8.1.: Server based Indoor Navigation System.

Together with developing theoretical algorithms for the enhancement of positioning accuracy, a real indoor navigation system has been developed for over a period of three years with the contributions from many employees and students at the Department of Communications Engineering, University of Paderborn. The system is the result of our attempt to bring the theoretical research results into reality.

Before building the indoor navigation system, one needs to answer the question: Which criteria must the system meet? To our understanding, the most important criteria to evaluate a real system are costs, ease of deployment, stability, security, scalability and maintainability. The better the criteria are fullfilled, the better the system is. System architecture has the most important impact on those criteria. As a consequence, to achieve the required criteria the best, different system architectures have been analysed. There exist two common architectures of positioning systems: First, smartphone based system in which all data are stored and processes are performed on the smartphone itself, e.g., the solutions from WIFARER [77],

without any communication with other devices. Second, server based system for indoor positioning purpose in which the tasks are seperated for server and clients, e.g., REDPIN [78], and information is exchanged via a communications protocol.

Recently, a project named "GreenPAD" which aims to save the power consumption of the WLAN infrastructure of the University of Paderborn has been reported [79]. Their proposed method basically tries to detect the unused APs to power them off, or measures the demand of clients to turn on or off a network interface. This might help to save energy consumption which is the goal of their project, however, it may strongly affect the performance of a WiFi fingerprinting based indoor positioning system. If they adjust the transmitted power to further improve their proposed approach, all WiFi indoor positioning systems will be destroyed or produce very unreliable results. While the problem of turning on and off the devices or network interface might be accounted for by our proposed advanced EM algorithm, problems caused by adjusting transmitted power remain unsolved for any WiFi positioning system. It is noted that other systems might not work well if any of those two problems occurs. However, as reported in [80], reducing transmitted power of a 802.11n Network Interface Controller (NIC) does not yield significant power savings. Yielding marginal power savings at the cost of a break down of the indoor positioning system may not be a good choice.

In the following, the advantages and limitations of a smartphone based and a server based system are discussed.

Smartphone based System

A smartphone based navigation system has several advantages. First, it is easy to develop since it mainly operates on the smartphones. Second, it is able to work at any time and in any area where the required data are available. And third, it is immune to the RSSI data mismatching problem described in chapter 5. These advantages come from the fact that the database is built and stored locally on each individual smartphones.

However, it has some major disadvantages: first, as the RSSI training data are collected by an individual smartphone and used for itself, it is not easy to use the data obtained by a specific smartphone to improve the positioning accuracy of the whole system. Second, map data might be unsynchronized among smartphones since they are stored locally. Third, upgrading positioning algorithm requires users to reinstall the application which is not very user friendly. And fourth, with some old or cheap smartphones of limited computational power, it might be difficult to apply complicated positioning algorithms which normally require high computational power to complete the calculation with a reasonable latency.

According to the advantages and the disadvantages as discussed above, smartphone based positioning system might be only suitable for a single user or a small number of users.

Server based System

A server based architecture, on the other hand, gets rid of all the weaknesses of the smartphone based system. Since the database, i.e., the map data and RSSI training data, are stored on the server, RSSI training data gathered by any individual smartphone can contribute to the improvement of the overall system performance, resulting in better positioning accuracy for all users. Map data are always synchronized among smartphones since they are distributed from a server. With a server based system, once the server is developed as the processing

center, it is also easy to develop any new and probably complicated algorithm without any serious issue since the computational power of a server is normally much higher than a smartphone.

However, this architecture also has some disadvantages. First, it requires an on-going network connection, which is not always available, for the communication (data exchange) between server and clients. Second, there is a need of a refining method to solve the mismatch of the data collected by different devices. Moreover, keeping the network connection at all times makes the application consume more battery power of the smartphones.

Fortunately, the first limitation can be solved by using the inertial sensor system which is present on smartphones. The smartphone navigation application is able to deliver positioning service in a short time period with a reasonable accuracy, when the connection is dropped, using dead reckoning techniques. The second limitation can be handled efficiently by applying the proposed model adaptation which was discussed in chapter 5. Power consumed by communication procedure and keeping a network connection is reasonable since it is much less than the power consumption of other tasks such as displaying.

Hence, a server based architecture might be suitable for an indoor positioning system with many users and large deployment area.

Our System

Since each architecture discussed above has its own advantages and disadvantages, a mixture of the two which combines both of them would be a good solution to keep the advantages and get rid of the disadvantages. Within this project, a server based architecture for indoor navigation was developed, where the server stores the database, i.e., the map data and Wi-Fi training data, and is the processing center. In the first generation of the system [73], the clients, i.e., smartphone application, take the resposibility of gathering data, such as WiFi data and inertial sensor information, as well as representing the user position on the smartphone screen graphically. Recently, the resposibilities of the clients in the system have been changed such that the clients do not only gather data but are also able to process the data to come up with position estimates. The changes were made because the WiFi connection between server and client was not very stable, especially at the areas with very few APs. Also the handshaking moment when the connection is handed over from one AP to the next, was problematic. These problems may cause pauses in displaying the map of the application which annoys the users.

The system can operate in two modes: online and offline. In online mode where internet connection is required, the client sends a request, i.e., positioning or navigation request, to the server and waits for the response containing the corresponding result. In the offline mode, the client can process the measured data by itself to come up with the final estimate of the user location or the navigation path if it has the local database. This helps to handle the problem of loss of connection which poses a significant problem in case of operating solely based on the server as the approach used in REDPIN. The local database can be acquired from the remote server and stored on the local storage of the smartphone to be used once the smartphone operates in the offline mode. It is guaranteed that the local database is almost always synchronized with the shared database stored on the server since it can be updated at any time by the user if an internet connection is available. By doing that, we avoid the database unsynchronization problem of the approach used in the WIFARER system where

local databases are created on the clients.

The smartphone application was developed to interact with the user via a user interface where the user can manage the operation mode of the application, i.e., application settings, localization task, routing task, and so on. With our developed system, it is very easy to apply any enhancement of algorithms for parameter estimation or changes in the map data since all the modifications are needed to be made on the server only.

In the following, the system architecture, i.e., the server structure, the smartphone application architecture and the communication procedure, as well as some main features of the developed system, i.e., data gathering, positioning and navigation, are presented. As mentioned above, the quality of the system depends on its properties such as cost, deployment simplicity, stability, security and scalability. Since our system is built upon the existing hardware, the requirements of a low cost and ease of deployment are met. However, building up the system requires a high effort for collecting training data which is an issue for any fingerprinting based indoor positioning system. The REDPIN system can be quickly deployed without high training effort as it asks the users to contribute training data to the server. If without any supervision, the system must consider all the received measurements to be reliable. This seems to be very sensitive to the behavior of the users. The database of the REDPIN system could therefore be destroyed any time by any user which is problematic. The other properties of our system are discussed in the following system description.

8.1. Overview of Server Architecture

The developed system contains two main components, the server and the clients. This section presents an overview of the server architecture which is given in Fig. 8.2.

The developed server is divided in three parts, the front-end HTTP server, the FastCGI module and the back-end. The front-end web server employs the lighttpd server [81]. This is a free open source web server which has been tested and proved to be stable, secure and flexible. It provides a stardardized Hyper Text Transfer Protocol (HTTP) which helps us to avoid spending effort on designing the communications protocol between server and clients. lighttpd is probably a good choice for the future since it is developed to optimize the performance of server regarding speed and load problems. It would not be an issue if many clients use the services of the system at the same time. Setting up the lighttpd server is straightforward, please visit [81] for more detail.

The web server receives requests from clients and forwards them to the appropriate FastCGI modules [82]. FastCGI, a variation on the earlier Common Gateway Interface (CGI) supported by lighttpd, is a binary protocol for interfacing the server application programs with web server. FastCGI has the following properties: first, FastCGI allows the server to handle more requests at once. Second, the FastCGI module can be implemented seperately by any programming language that supports network sockets such as C, Java or Python. And third, the FastCGI modules run as seperate processes. The reasons that FastCGI was chosen are because of the following properties:

- In our previous work as described in [73], a positioning server was developed by ourselves using C++ programming language where many functions have been written for database access, position estimation, navigation path estimation, and so on. Lately, as

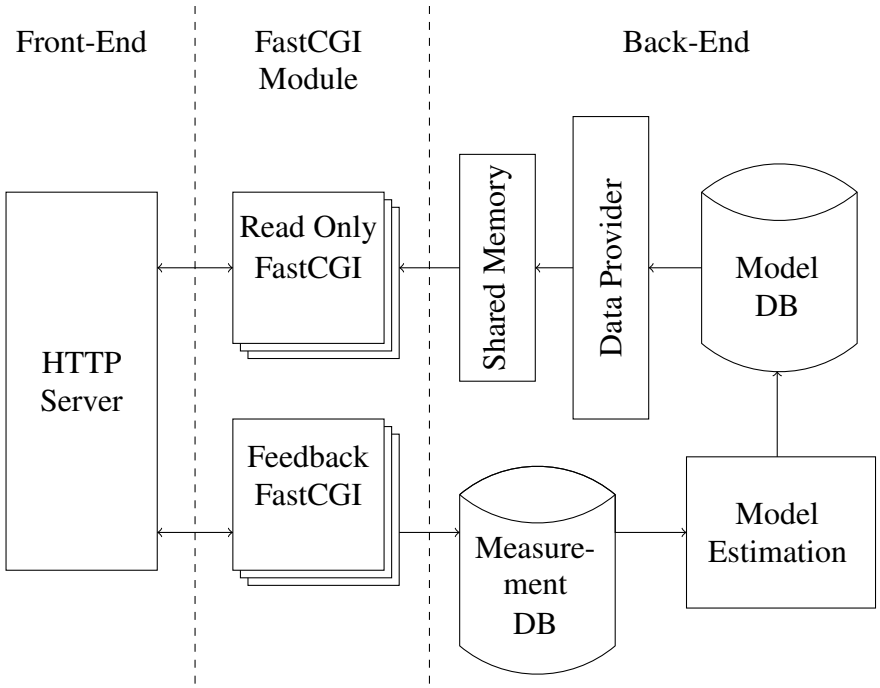


Figure 8.2.: Server architecture of the indoor navigation system.

we decided to use the lighttpd server, we needed to write the server application programs. FastCGI is a good candidate since FastCGI modules can be implemented in C++ which simplifies the procedure of migrating the already implemented functions in the old server to the new one.

- The old server has a severe problem that if any process crashes, the whole system stops working. By using FastCGI in the new server, the problem is solved. As the FastCGI modules run in seperate processes, only the crashed process stops working while the other processes in the system still work. This makes the system more robust and flexible because developing and testing any module on the server does not affect the operation of the other stable modules.

In order to process the requests, the FastCGI modules need to access the database in the back-end. There are reading and writing access tasks corresponding to specific requests. The writing-access modules are needed to accommodate new data in the database, for example, when the request is "SetFingerprintRequest" which asks the server to add new measured data to the database. The reading-access modules, which are the corresponding modules for the majority of requests, i.e., positioning request or navigation request, operate on the estimated training models which are stored in the model database.

In the back-end, the RSSI database was seperated into 2 sub-databases: the measurement database which stores all raw RSSI measurements, and the model database which stores the model estimated from measured data in measurement database. The model estimation program where the proposed EM algorithms are implemented is used to estimate training model from measurement data. At the moment, this program is activated by the system manager

regularly. It is not necessary to reestimate training models every time a new measurement is added to the measurement database since the changes of the estimated parameters are negligible. As a result, new measurements are added to the measurement database every time the server receives the "SetFingerprintRequest", but model estimation is only performed regularly at a longer interval, e.g., every week. This architecture of the back-end supports the possible future feature of learning the models online such that the training database is updated during the usage by customers.

To speed up the processing time of the data reading procedure, it was decided to use shared memory since it helps to significantly speed up the data reading procedure. "DataProvider" is a C++ program which simply reads the data from model database and writes them to the shared memory in an appropriate format which helps to optimize the positioning and navigation procedures.

8.2. Database

The indoor navigation database is part of the back-end of the server. In our system, the raw RSSI measurements, the map information and the estimated models are stored in the PostGIS database [83] which is an extension of the PostgreSQL database [84].

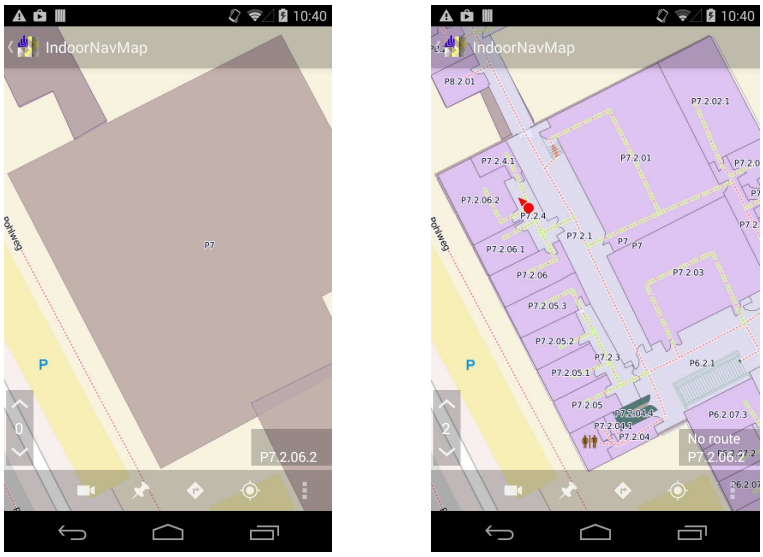
8.2.1. Map Tile Data

First of all, to represent the user position in an indoor environment, the map tiles need to be created. After a careful survey of map creating tools, "mapnik" [85] and its associated tools, e.g., JOSM [86] and "osm2pgsql" [87], are chosen to implement the map creation procedure [88]. The reasons for choosing these map editing tools are, first, maps can be edited offline and stored on a standalone computer which is important for security reasons, and second, map data can be rendered with multiple zoom levels and configurable properties, i.e., colors, which are very important for the quality of the map displayed on the smartphone screen. The map files to be edited are originally downloaded from the OSM server, then with the use of the JOSM tool, the details of the indoor floor plan can be added. Each of the edited OSM map files containing the information of one floor level of all buildings within the university is imported to a PostGIS database using "osm2pgsql". The information in this seperate database is then used to render the map tiles of the corresponding floor levels using "mapnik". The map tiles are stored in the server as "/Floor/z/x_{tile}/y_{tile}.png", where "Floor" is the name of the rendered floor level, z is the zooming level, x_{tile} and y_{tile} are the integer indicators of a map tile which can be computed from the specific longitude, latitude (lon, lat) coordinates using the following equations

$$x_{tile} = 2^z \frac{(lon_{deg} + 180)}{360},$$
$$y_{tile} = 2^{z-1} \left(1 - \frac{\log(tan(lat_{rad}) + cos^{-1}(lat_{rad}))}{\pi}\right),$$

where the subscripts deg and rad indicate the units of the lon and lat are degree and radian, respectively.

Fig. 8.3(a) shows a smartphone screenshot when the original OSM map without any detail of the floor plan is displayed. After editing with JOSM and rendering with mapnik, the detail information of indoor map is added. As a result, the map with room, corridor, stair, and so on, is displayed as shown in Fig. 8.3(b). Fig. 8.4 shows a snapshot of the JOSM environment which is used to edit OSM map files. As can be seen, plenty of objects are added to the OSM map file. Since the map is edited manually, map creating procedure is a very time consuming task for a large deployment area. Since no modification of the standard rendering procedure with mapnik is made, except for some definitions for map rendering boundaries and colors, the details of the rendering procedure are skipped in this report. More information can be found in [88] and on the website of the mapnik project [85].



(a) Original OSM map (b) Edited map

Figure 8.3.: Examples of original map and edited map

Figure 8.4.: Map editing with JOSM

However, since the map information is stored in seperate Geographical Information System (GIS) database for each floor, for multi floors navigation purposes, calculating the navigation path by searching across multiple databases is inconvenient. To solve this problem,

markerid
lon
lat
room
layer
ref

markerlocations

connectionid
markerid
linkedmarker
distance
highway
access

markerconnections

Table 8.1.: Map information tables in the database: storage of the geographical information of fingerprint positions, and the information about the direct connection between any pair of positions.

we decided to combine all the seperate OSM map files into one global OSM map file. The navigation database is then built up by importing the information from the global map file to the PostGIS database.

The rendered map tiles are stored on the server and delivered to the clients for map displaying purposes once the server receives map requests from clients. Map information stored in the navigation database is the information of fingerprinting positions and the connections among those. This information is gathered from the standard tables by a C++ program and stored in two new tables (see Table 8.1 for an illustration). The bold fields are the so-called unique keys or master keys of the tables. Table `markerlocations` contains all the information about any fingerprinting position such as longitude, latitude, room number, layer (floor level) and ref (corridor, toilet, ect.). These fingerprinting positions are the positions where training data will be collected. It is very convenient to distribute the training positions during the map editing procedure since one can choose the critical and interesting points on the maps to mark them as fingerprinting points. It is also easy to manage the density of the fingerprint grid. Table `markerconnections` contains the information about the direct connection between any pair of positions, e.g., ID of current considered position (markerid), ID of the connected position (linkedmarker), spatial distance between these positions (distance), type of connection (highway), i.e., footway or bicycle, and accessibility information (access) to indicate whether the connection is blocked or if there is free access. The information in Table `markerconnections` is needed for the route estimation procedure and the employment of the HMM which was discussed in Chapter 6.

8.2.2. RSSI data

RSSI data are divided into two types: raw data (RSSI measurements) and processed data (the estimated training models). Table 8.2 shows the tables in the database which store the measurement data. There are several reasons for storing the raw data. For example, raw data can be used for future research, i.e., if new algorithms are used to estimate the training model from measured data, or for comparison purposes, i.e., evaluating the positioning accuracy of other algorithms on the same data set. By storing the raw data in the way as described in Table 8.2, it is possible to re-generate the original measurements. Tables `scanresults`, `measurements` and `hardwareinformation` are used together to store the RSSI measurements, the position, ID and orientation of the device, and the time stamp of the measu-

scanid
measurementid
mac
rssi

scanresults

measurementid
lon
lat
layer
hwid
orientation
measurementtime

measurements

hwid
modelid
uniqueid

hardwareinformation

Table 8.2.: Measurement tables in the database: contain the RSSI measurements, the position, ID and orientation of the device which has been used to collect data, and the time stamp of the measurements

parameterid
markerid
macid
numberofmeasurement
numberofobservation
mean
variance
rate
sumofobservations
sumofsquare
updatedtime

parameters

macid
bssid

macaddresses

Table 8.3.: Training model information tables in the database: contain the estimated parameters and the sufficient statistics information.

rements. The information about the smartphone hardware is stored to support the adaptation procedure as described in chapter 5.

In addition to storing the raw data, the RSSI database also contains the tables to store the information of the estimated training models as well as the sufficient statistics information as depicted in Table 8.3. Table `parameters` stores the estimated parameters as well as the sufficient statistics information. The estimated parameters are the means, variances and rate (dropping rate) of the observed RSSI data of the AP, specified by macid, at a location, specified by markerid. Sufficient statistics parameters such as number of measurements, number of observations, sum of observations and sum of observations squared, are used for incrementally updating the training model to reduce the estimation time using the EM algorithm. Obviously, estimation time is not a serious problem when the amount of training data is small. However, training data are collected over the course of time and model estimation from scratch would take much more time than applying the incremental update method. For convenience of AP searching, the table `macaddresses` is created to store the MAC addresses of all the observed access points.

8.3. Shared Memory

As mentioned above, shared memory is used to represent the database in order to speed up the data reading procedure. The structure of the data stored in shared memory is similar to the structure used in the PostGIS database as presented in the previous section. Seperate shared memory areas are created to store the information of fingerprinting positions, connections, MAC addresses and trained models. In addition, to speed up the localization procedure, other areas are created to store the information of the pre-computed positions at which a given MAC address was observed in the training data. The reason is that in fingerprinting based positioning, online measured data are compared with the training data in the database to come up with the decision of the user position estimate. Obviously the bigger the database becomes, the more computation time is needed to complete the comparison procedure. To get rid of this problem, instead of computing the similarity between the online measurement with the whole database, the localization module has to compute it at the possible positions only, where at least one of the APs, which is present in the online measurement, was observed in the training data.

8.4. Overview of Smartphone Application

This section presents an overview of the smartphone application. Although the developed application is able to support many tasks, e.g., data gathering, localization, navigation, social features (peer group finding) and 3-D representation of the routing path, in the following, only the structure which is related to the main features, which are localization and navigation, is presented.

Fig. 8.5 gives an overview of the reduced version of the smartphone application architecture as a class diagram.

It should be noted that in Fig. 8.5, the details of each class, i.e., variables and methods, are not presented since this would extremely extend the size of the graph. In the following, the role of each class in the application and the relation of classes will be summarized.

As can be seen in Fig. 8.5, besides the regular classes, there are several special types of classes in an Android application such as activity, service, application which operate differently [72] as summarized below:

- *Activity*: An activity is an application component that provides a single window with a user interface that interacts with the users in order to do an action. An application may consist of several activities to handle different tasks. An activitiy can start another activity. Once a new activity starts, the previous activity is stopped. The system preserves the previous activity in a “last in, first out” stack. Therefore, when the user is done with the current activity and presses the “Back” button of the smartphone, the previous activity will resume.

- *Service*: A service is an application component that can operate in the background and does not provide a user interface. A service can be started by another application component, i.e., activity, and keeps running in the background even if the user switches to another application. Several types of service implementation can be used to manage

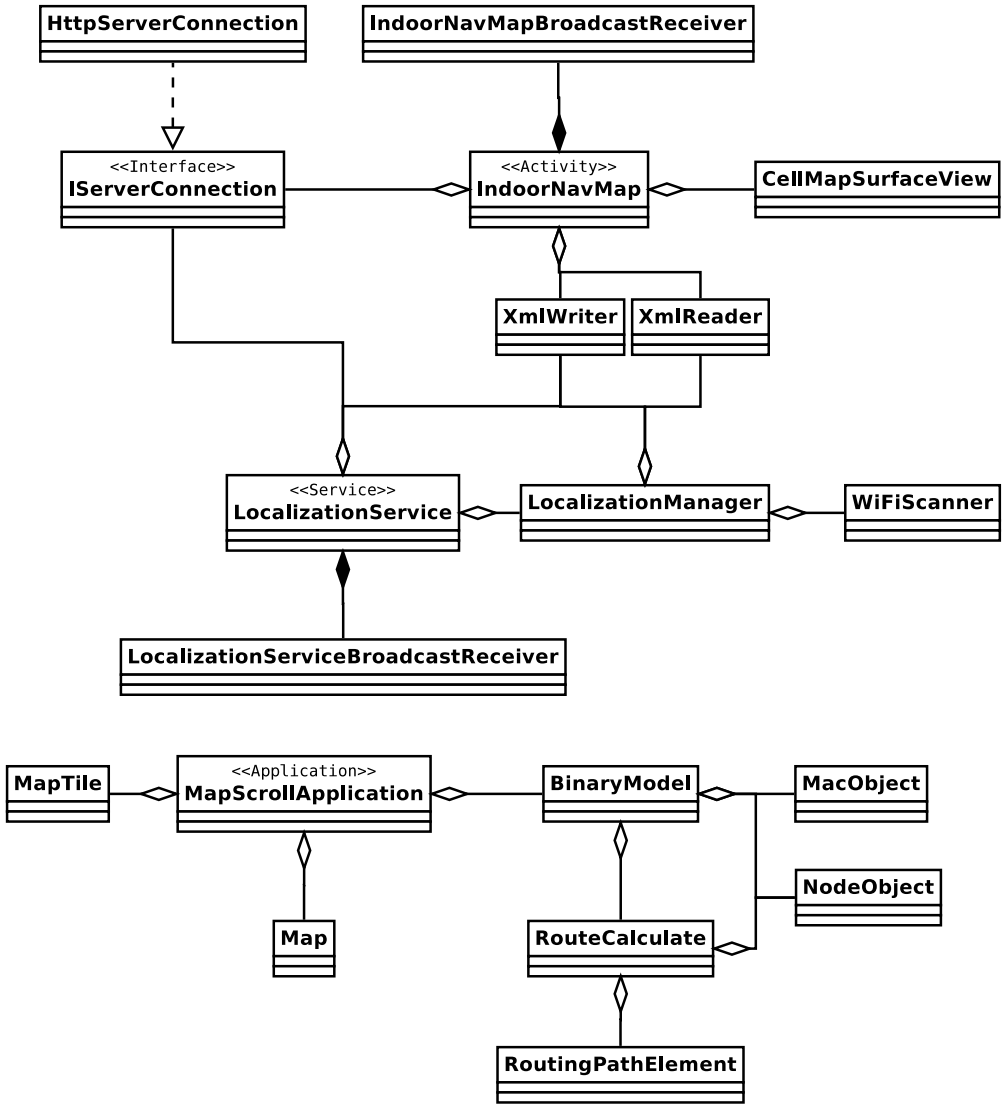


Figure 8.5.: Class diagram of Java application architecture

the lifecycle of a service that the service may stop itself when it completes the task or its binding component stops or an activity stops it.

- *Application*: An application is a base class to maintain global application state.

In our application, the main activity is "IndoorNavMap" which is launched when the user starts the application. All other activities, i.e., WiFi scanning, setting, etc., can be activated from this activity, see the manifest file of the application in Appendix A.3.1 for information of all activities. Fig 8.6 shows a snapshot of the application when it is launched by the users.

In the following discussion, the relation between the classes in Fig. 8.5 and the role of each class are summarized:

- "IndoorNavMap" uses the setting information and initialized information stored in "MapScrollApplication" and the appropriate methods implemented in "CellMapSurfa-

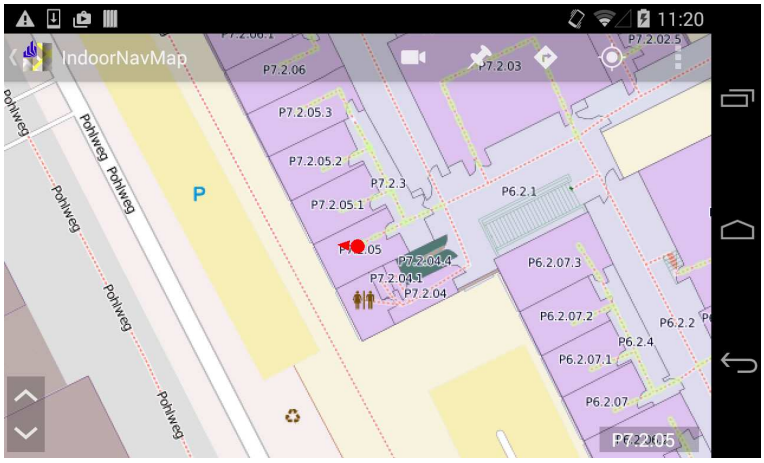


Figure 8.6.: Java application showing floor plan information and user position (red dot).

ceView" to display map and other information such as current user position, navigation path, etc. .

- "MapScrollApplication", the application class, is implemented to maintain all the global states and the data which are needed for map displaying, offline localization, navigation, and so on. For map drawing, positioning and navigation, "MapScrollApplication" stores the metadata of map data, i.e., objects of "Map" and "MapTile" classes, and the information for localization and navigation purposes, i.e., "BinaryModel", "MacObject", "NodeObject" and "RoutingPathElement", as descibed in section 8.2.

- "CellMapSurfaceView" extends the "SurfaceView" class provided by Android APIs, and contains all methods to manage map drawing and user interaction.

- "IServerConnection" is the interface declaring all the methods which are implemented in "HttpServerConnection" to support the communication procedure between the application and the server. These methods can be called by "IndoorNavMap" for data downloading or by "LocalizationService" in case of doing localization in online mode.

- As all the messages are in the pre-defined XML format, methods for writing/parsing data to/from XML format messages are implemented in "XmlWriter" and "XmlReader" for communicating with the server.

- "LocalizationService" is the service for handling the tasks related to the localization procedure. This service is started by "IndoorNavMap" and calls the methods implemented in "LocalizationManager" to perform localization. This service is stopped automatically by the system once the user terminates the application.

- "IndoorNavMapBroadcastReveiver" and "LocalizationServiceBroadcastReceiver" are implemented to support the data exchange between "LocalizationService" and "IndoorNavMap". These two classes extend the "BroadcastReceiver" provided by the Android APIs.

- "WiFiScanner" is responsible for WiFi data acquisition for localization procedure.

- "RouteCalculate" contains all the methods related to navigation path calculation. These methods are activated by "IndoorNavMap" via "MapScrollApplication" where the information of the calculated route is stored. By doing this, if the user switches to any other activities or applications, once user gets back to the "IndoorNavMap" activity, the navigation path will be displayed without re-calculation. More details about the navigation procedure will be presented in section 8.5.3.

8.5. System Operation

This section presents the operation of the main features of our indoor positioning system from a user's view, as well as the flow sequence of the smartphone application for each feature. The developed indoor navigation system is able to support several services such as gathering of training data, localization, navigation, some initial versions of social features such as peer group finding, and 3-D representation of the routing path. The main features, i.e., gathering of training data, localization, and navigation, are discussed in detail, a short introduction about the other features is given at the end of the section.

8.5.1. Data Gathering

This feature is responsible for gathering training data. To do this, the smartphone application performs WiFi scans and writes the measured data to a "SetFingerprintRequest". It should be noted that on the smartphones, at the moment, this feature is only available when the application runs in the development mode in order to avoid problems in case someone tries to harm the database. In future, for extending the system towards online learning, this feature could be performed in the background of the smartphone application during the usage by users. The development mode is activated by the developers by choosing the option "Activate developer view", as shown in Fig. 8.7.

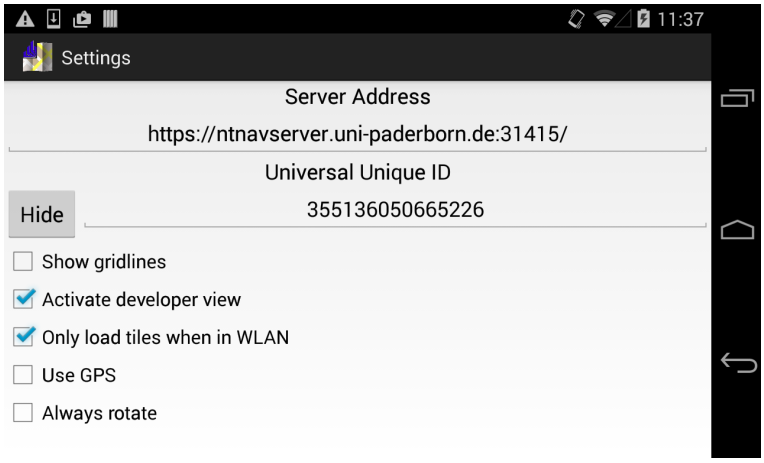


Figure 8.7.: Java application showing setting options.

The development view of the smartphone application is illustrated in Fig 8.8. In the figure, the red circles indicate the positions at which RSSI training data have already been collected.

By doing so, the developers can then choose the fingerprint positions which have not been trained to collect RSSI data. The red "+" sign on the map shows the position where WiFi data are going to be collected. The developers can modify the position of the red "+" sign by moving the map. To collect the RSSI data, since each position needs a sufficient amount of measurements to accurately estimate training models, an activity named "WLANDataAssemblerActivity" was developed to support this requirement.

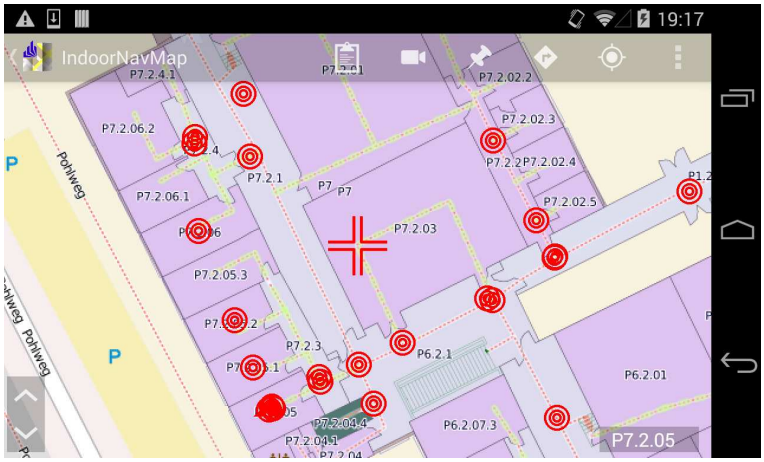


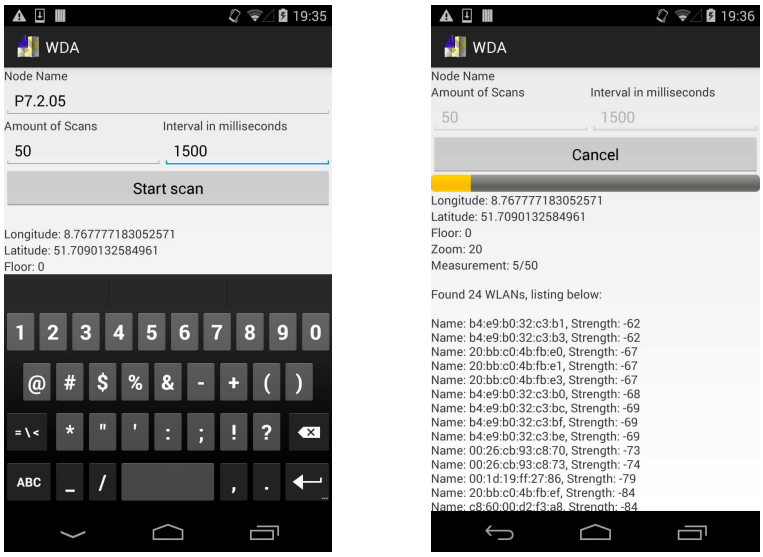
Figure 8.8.: Java application showing the development mode: the red circles are the positions where training data were already collected, the red "+" sign on the map shows the position where WiFi data are going to be collected.

Once the developer starts the data gathering procedure, a window appears which allows the developer to verify the information of the current position and enter additional information, i.e., name of the position, amount of scans and scan interval, see Fig. 8.9(a). Depending on the sampling rate of the WiFi sensor of the test smartphone, the scan interval need to be large enough to avoid duplication of the measurement data, We observed the sufficient scan interval of about $1{,}5\,\mathrm{s}$ for Samsung smartphones, while for a Sony device it is roughly $5\,\mathrm{s}$. Once the information is validated and entered, the measurement process will be started by simply clicking the "Start scan" button. Fig. 8.9(b) shows the screenshot when the smartphone performs a WiFi scan, where the number of measurements and scanned data are shown.

The measured data are then written into an XML file which is stored on the local storage of smartphone once the scanning procedure is completed. The XML file contains the measured RSSIs, MAC addresses of the observed APs, the smartphone information, as well as the information of the location where the measurements are taken. It should be noted that each XML file contains the training data at one position only.

After the measurement campaign, all XML files are imported into the database on the server using a C++ module named *setfingerprint*. This module parses the request and accesses the PostGIS database to insert the measured data. For each file, *setfingerprint* produces a response message to inform whether the data are successfully inserted to the database or not. Appendix A.4.1 shows an example of the "SetFingerprintRequest" and response messages.

Our system is also able to support the online mode of the measurement campaign, i.e., the measured data are sent directly to the server after each measurement via a wireless network,



(a) Setting for WiFi scan (b) WiFi scanning

Figure 8.9.: WiFi training data gathering

if we use the *setfingerprint* module as a FastCGI module. This option provides the possibility of gathering data during the usage of users for an online learning approach. However, at the moment, no algorithm has been developed to measure the reliability of a measurement. Therefore, we temporarily disable this option to avoid accidental measurements containing wrong information which may harm the database. The indoor environment and WLAN network are subject to change over time, this requires regular database maintenance (update) to keep the positioning accuracy. It is infeasible to retrain the database after every change of the environment or the network infrastructure. The reason is collecting training data is very time consuming, especially for the large deployment area. Therefore, a method for automatically updating database, i.e., the online learning approach, could be a solution.

The sequence diagram of Fig. 8.10 shows how the smartphone application performs WiFi scanning in offline mode, i.e., the scanned data are written in an XML file and stored in the local storage of the smartphone.

8.5.2. Localization

For localization, users can choose either the offline localization mode or the online localization mode in the settings of the smartphone application . In offline mode, position estimation is performed locally on the smartphone using the database which is stored on its local storage without any need of internet connection. In online localization mode, which requires network connection for data exchange, position estimation is performed on the server. Figure 8.11 shows the options for localization, i.e., operation mode and the data source.

The sequence diagrams shown in Fig. 8.12 and Fig. 8.13 present the communication among the components of the smartphone application to perform localization in offline mode and online mode using WiFi information, respectively.

To perform the localization using WiFi data, smartphones periodically log the WiFi data which are the MAC addresses of the observed APs and their measured RSSIs. An estimator is then used to process the scanned data to determine the user positions In offline localiza-

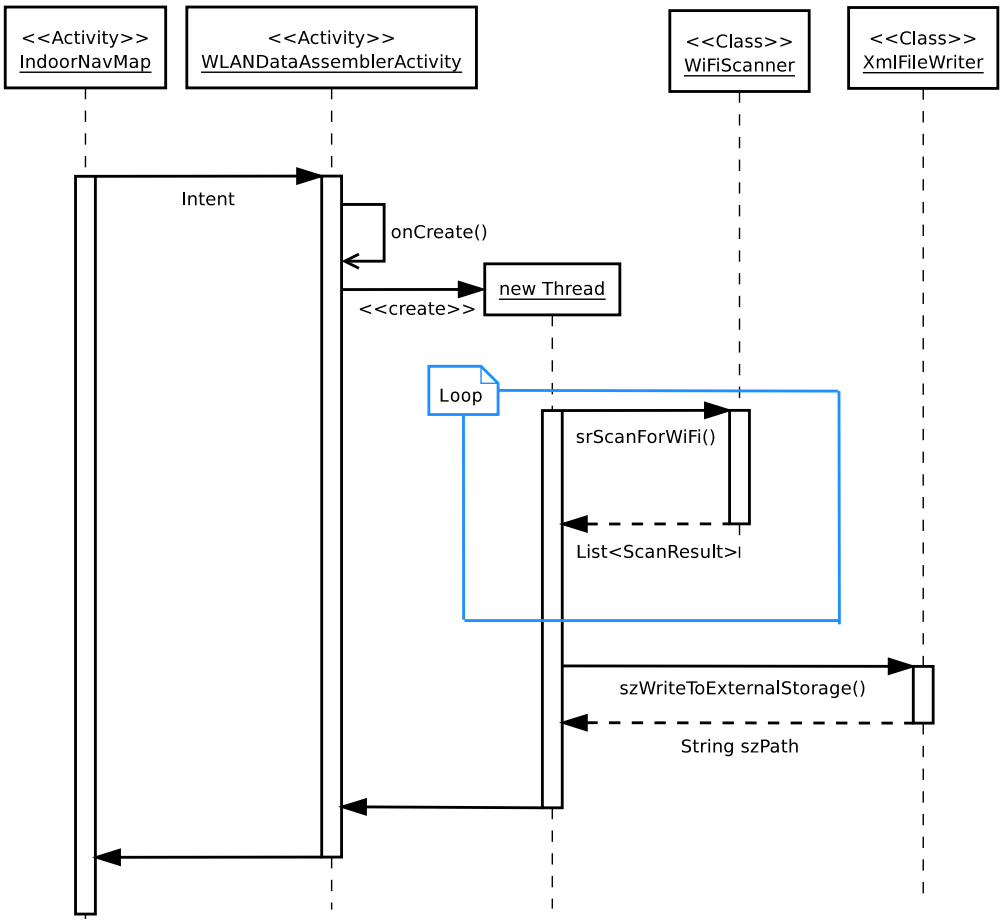
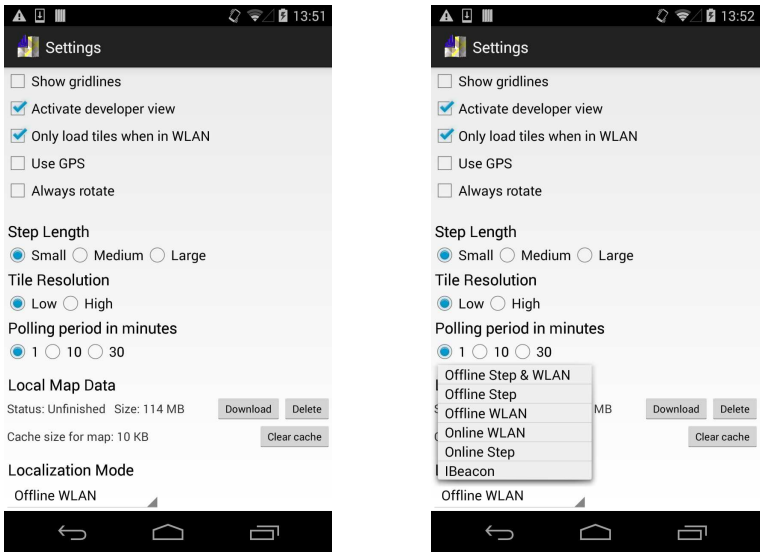


Figure 8.10.: Sequence diagram showing the data gathering procedure.

tion, the estimator is implemented as a method in the smartphone Java application which uses the local database to compare with observed data. Several issues may arise in offline localization, for example, the local database or the positioning algorithms might be out of date, as discussed in the beginning of this chapter. To solve this problem, the local RSSI database can be reloaded from server regularly, i.e., once a week, automatically or manually if network connection is available. The problem with positioning algorithms is unsolvable unless the user updates the application. Within this work, an automatic update procedure for smartphone application has not been developed yet.

For online localization, scanned data are sent to the server via the WLAN connection in an XML format. On the server, a FastCGI module named “positionestimate” was developed to parse the request and perform the localization procedure. The position estimation algorithm was implemented using the classification rule which was discussed in Chapter 4, the same algorithm is used for offline localization. This module uses the training models stored in the shared memory to carry out the localization procedure.

Sensor fusion for the improvement in positioning accuracy as discussed in Chapter 6, is not implemented on the server side, but on the client side. The reason is that the server must be a stateless server since there would be an explosion of the amount of HMM states which need to be stored if many clients request the localization service at the same time using RSSI



(a) Setting options (b) Localization modes

Figure 8.11.: Setting screen showing options for localization.

and step detection information. This would dramatically degrade the response time of the server.

Once the position estimate is obtained, a localization response is sent from the server to the client which contains the information of the estimated user position. The client parses the response and shows the user location on the smartphone screen, see Fig. 8.6 where the red dot represents the current user position.

8.5.3. Navigation

For navigation purpose, the user can input the information of the source and destination positions, i.e., room numbers, and the options for the expected route, i.e., using elevators or stairs. Fig. 8.14 shows an example of a smartphone screen when the user starts the routing function of the smartphone application.

Navigation, similar to localization, can be performed in either offline mode or online mode. In online mode, a navigation request with the information inserted by the user is generated by the smartphone application and sent to the remote server. On the server, a FastCGI module named “routeestimate” was developed which is able to parse the request and perform the route calculation. Data for the route calculation process are obtained by reading the shared memory. To perform the path search, several path finding algorithms have been considered such as the well known Dijkstra’s algorithm, the A* algorithm (a variant of Dijkstra’s algorithm) and the jump point algorithm. In navigation, the path cost is simply the geographical distance of any pair of connected positions. While Dijkstra’s algorithm examines all nodes to find the shortest path between the source and the destination, the A* algorithm is trying to examine the nodes which are potentially on the shortest path first to optimize the computational demand. This is the reason why the A* is also called goal-oriented Dijkstra’s algorithm. However, the A* algorithm needs heuristic weights to guarantee the solution is the shortest path. The jump point algorithm tries to optimize A* in case of uniform-cost grids, which is normally not suitable for an indoor environment since distributing the indoor fingerprinting

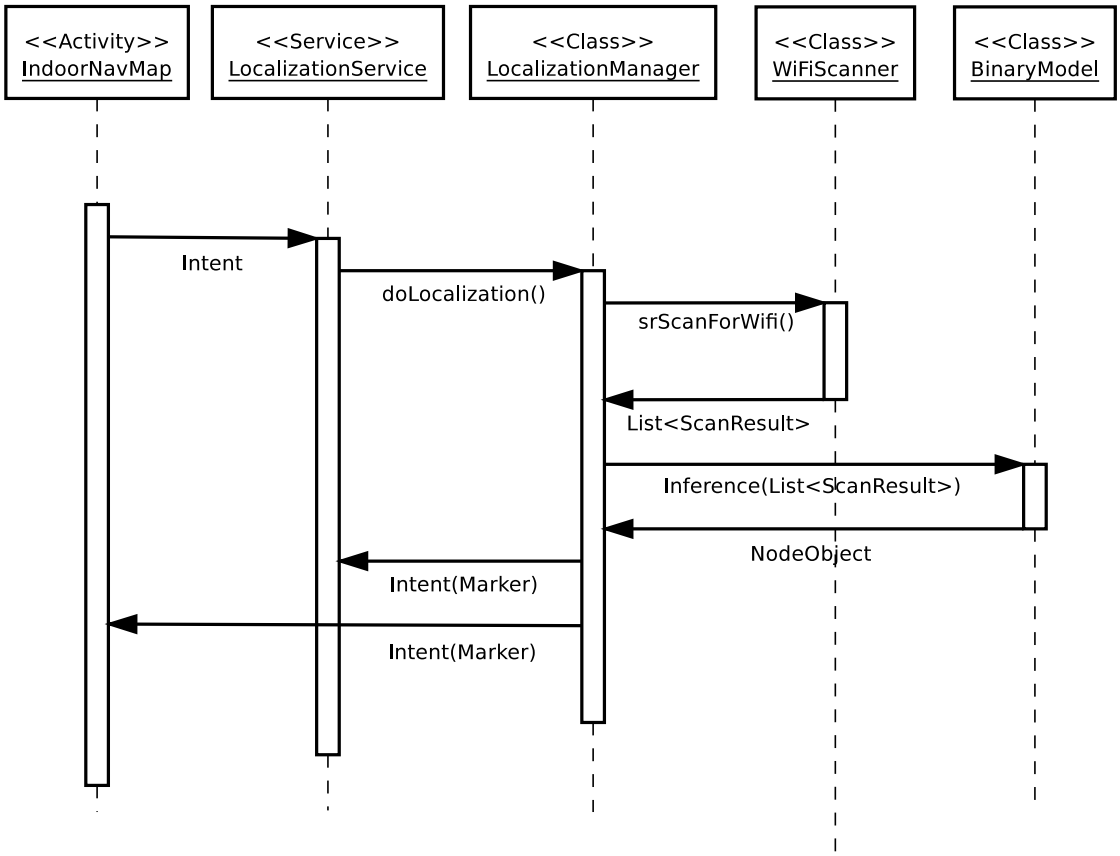


Figure 8.12.: Sequence diagram presents the offline localization procedure.

nodes uniformly is very difficult, probably not possible. Because of that, Dijkstra's and A* algorithm have been implemented to compare their performance, i.e., computational time and route estimate. Using the map data of almost the whole University, these two algorithms are able to finish the calculation in a very short time and the difference of the calculation time is unnoticeable. While the result of Dijkstra's algorithm is always the shortest path, A* sometimes provides a suboptimal solution since its performance depends on the heuristic weights. The larger the weights, the faster the algorithms is, however, the larger the risk to find not the optimal solution. As a consequence, Dijkstra's algorithm is chosen for our system.

Another reason for choosing Dijkstra's algorithm is the following: For the navigation feature, we would like to, first, display the shortest complete path between the source and the destination. Second, during the movement of the user, the current user position will be considered as the source of the route to adjust the guided path. In reality, the user is likely to move away from the guided path at some time and at some areas. For this case, a new route needs to be computed to guide the user from the current position to the expected destination. It means that the path is not fixed but changing over time according to the current position of the user. This is similar to how the car navigation works: if the driver drives the car away from the guided path, a new route is needed to guide the driver to get to the destination. To avoid re-calculating the routing path all the time, we modify the method of implementing Dijkstra's algorithm in a way that the algorithm starts from the destination and stops when

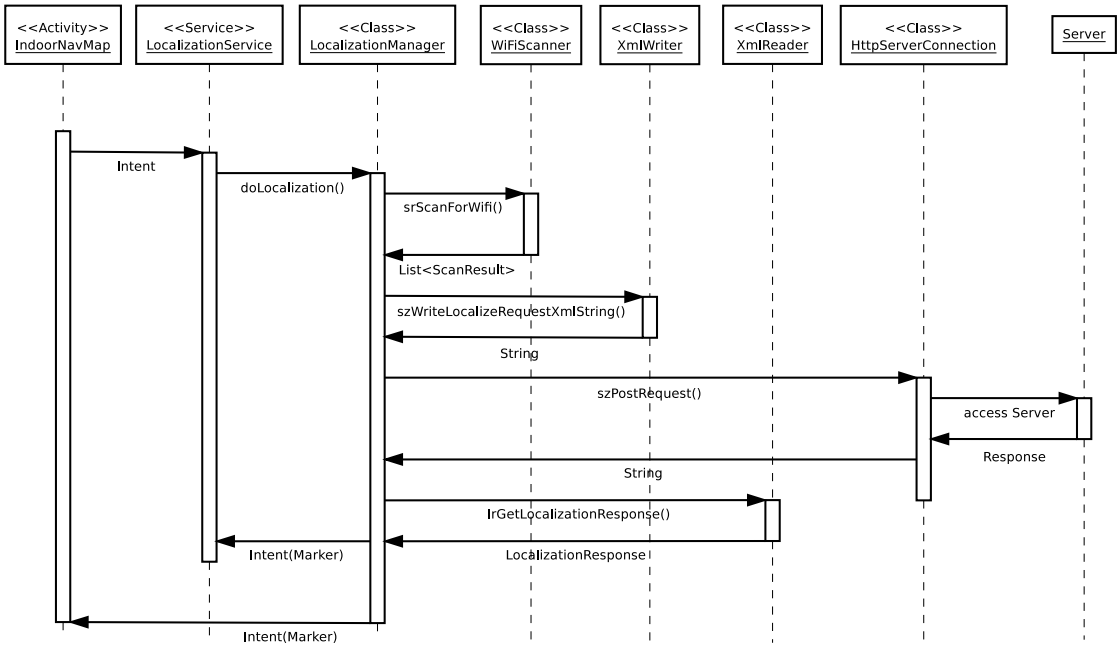


Figure 8.13.: Sequence diagram presents the online localization procedure.

all the nodes are examined. The calculated information is then stored temporarily during the current navigation process until the user terminates the navigation process. By doing that, everytime the displayed route needs to be updated, the current position of the user is estimated and the new path can be calculated quickly using the stored information. Note that A* cannot be used for this option since it is not designed to examize all the nodes but only for finding path between one pair of nodes.

The sequence diagram shown in Fig. 8.15 presents the communication among the components of the smartphone application to perform navigation.

Once the routing path is estimated a navigation response which contains the list of the positions forming the path is created and sent to the client. The client parses the response and displays the path on the smartphone screen, see Fig. 8.16.

To navigate through the university which includes multiple buildings, the system is able to provide outdoor navigation in addition to indoor navigation, as illustrated in Fig. 8.17. The green line indicates the indoor path while the blue one indicates the outdoor path which together form a complete path to navigate from one position in a building to another position in another building. In the outdoor environment, the position of the user is determined by using GPS information.

8.5.4. Features Under Development

At the moment, there are several features which are under development, i.e., peer group finding, 3-D representation of the routing path and iBeacon based localization. Since these features are in the testing mode, in the following, a brief introduction about them is given.

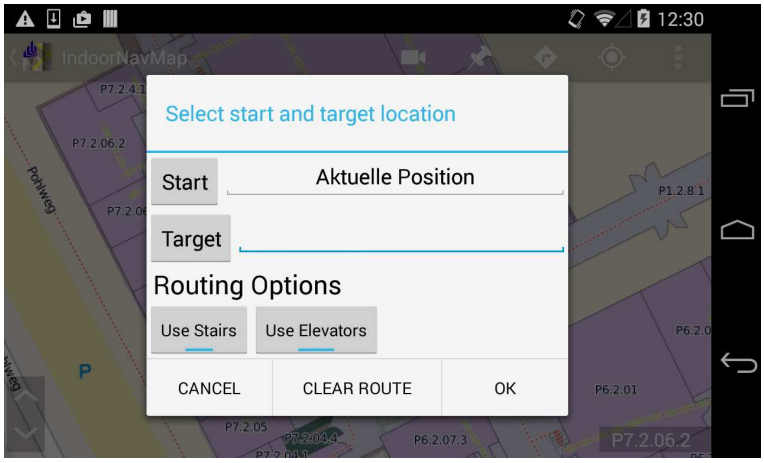


Figure 8.14.: Java application showing the routing options.

Peer Group Finding

Peer group finding, such as learning group finding, helps students to search for other students who have the same study interest to form a learning group. According to our opinion, this is an interesting feature of the system for students since in daily tasks, once the students are familiar with the university, positioning and navigation features are not really needed, except for emergency cases. As a consequence, social features like peer group finding help to keep the system alive. To use this feature, the user must create an account and register with the system. Every time the user wants to find the interested group, he must log in the system with the registered account. Users can choose to share their current positions with others. They can also decide not to do so if they do not want to share their current positions. Fig. 8.18 shows the screenshots when the user starts the peer group finding feature and after logging in the system.

3-D Representation of Routing Path

3-D representation of the routing path gives the users a nice overview of the path that they would need to follow in order to travel to the desired destination. At the current state of the system, the 3-D video is played once the user activates it, showing the 3-D path from the source to the destination. Fig. 8.19 shows some screenshots of the application when performing the 3-D representation of the estimated routing path. According to our opinion, this is a very interesting feature of an indoor navigation system since it shows the path and the nearby objects in detail. In the university this feature might be not very impressive. However, once the system is deployed in a museum, for example, the visitors can make a virtual tour around the museum before deciding which areas they will visit. In the future, this can be developed to show the 3-D view of the path following the movement of the user which is more user friendly. One challenge with this feature is that building the 3-D database is very time consuming. This feature was developed under a project which was granted by the University of Paderborn by combining the work of our department with that of another group which takes the responsibility for creating the 3-D map database. At the moment, 3-D map database contains the information of the main buildings of the university, the data for the

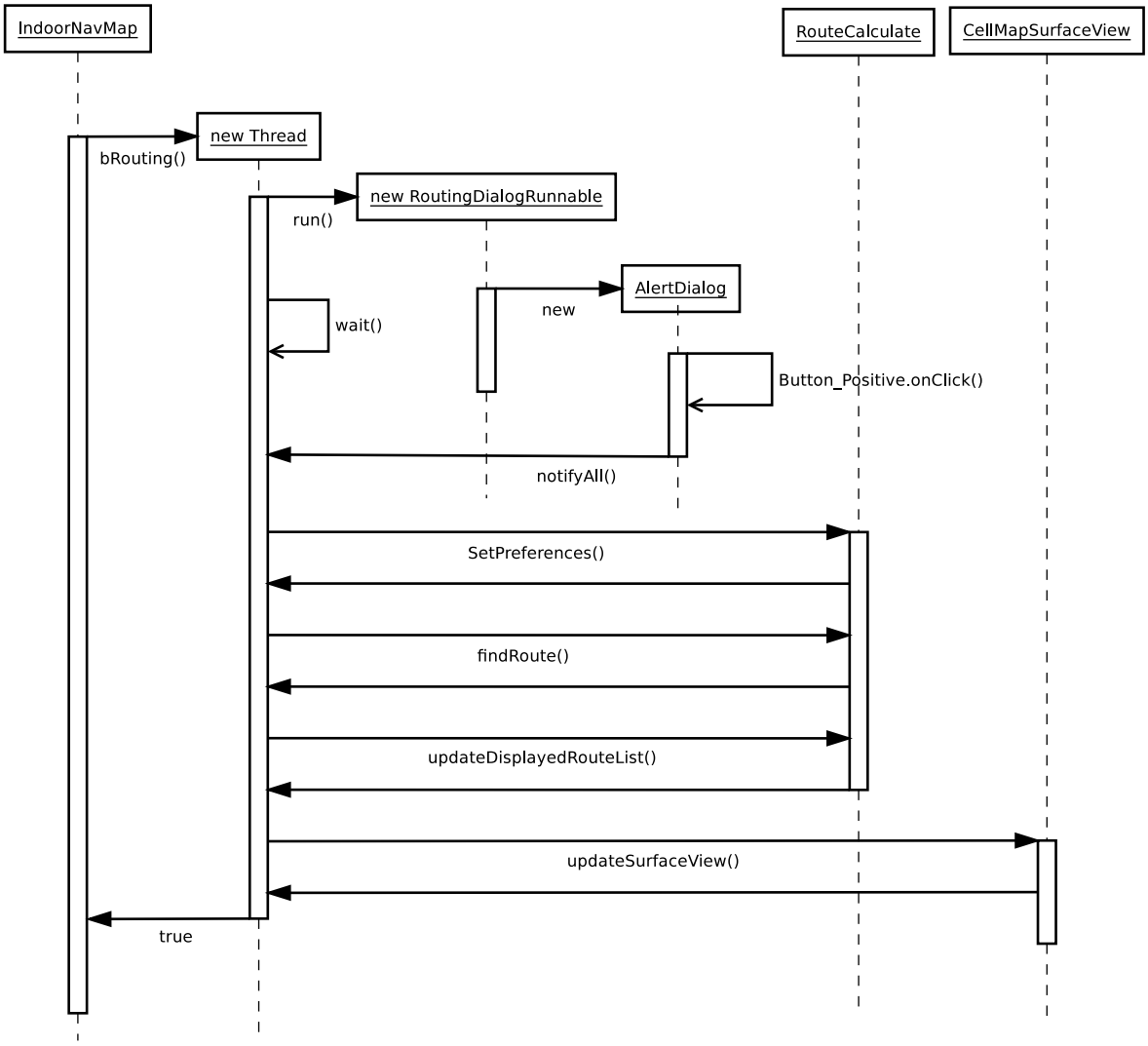


Figure 8.15.: Sequence flow presents the offline navigation procedure.

other parts are still under development.

iBeacon based User Localization

The iBeacon technology works using the Bluetooth Low Energy (BLE) technology which intends to provide considerably reduced power consumption compared to WiFi or ordinary Bluetooth. User location is obtained by applying the proximity technique using iBeacon information. To do localization, the smartphone continuously scans the iBeacon advertisements and determines the user position as the position of the iBeacon transmitter which has the strongest measured signal strength, assuming that the locations of iBeacon transmitters are known. It should be noted that this feature works only on the smartphones with Bluetooth Core Specification Version 4.0 (Bluetooth v4.0) or higher.

Since the proximity technique requires low computational demand, the positioning procedure using iBeacon information is performed on the clients. The position information of the iBeacon transmitters, i.e., positions, transmitted power, is stored on the server and smartpho-

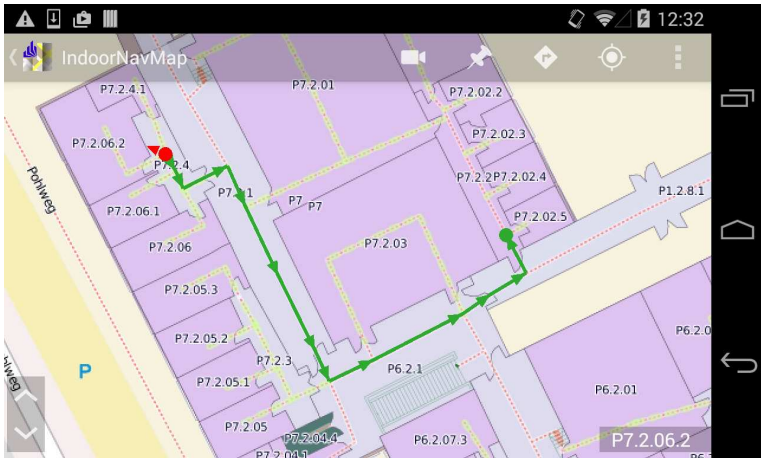


Figure 8.16.: Java application showing the indoor routing path.

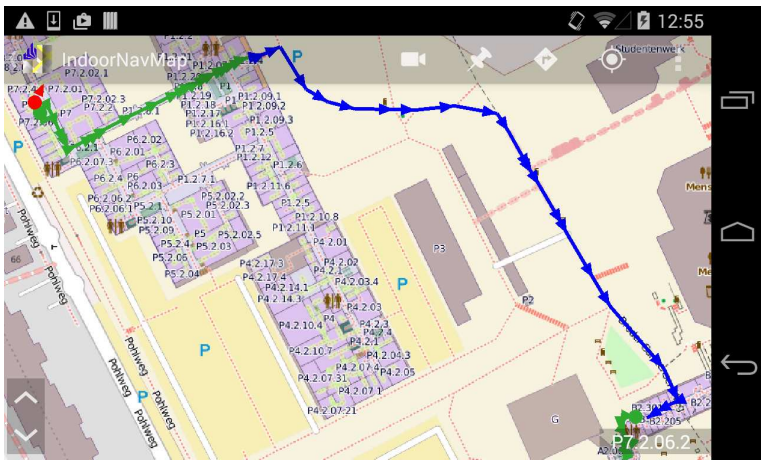
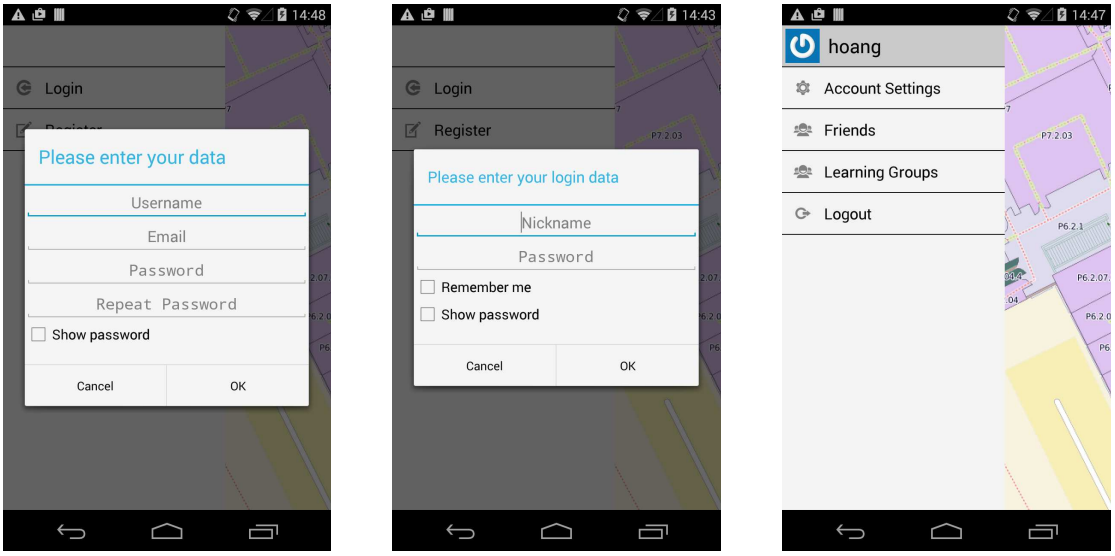


Figure 8.17.: Java application showing the indoor and outdoor routing path.

nes can update that information regularly. This is to ensure that the smartphones always have the same up-to-date information of the iBeacon system.

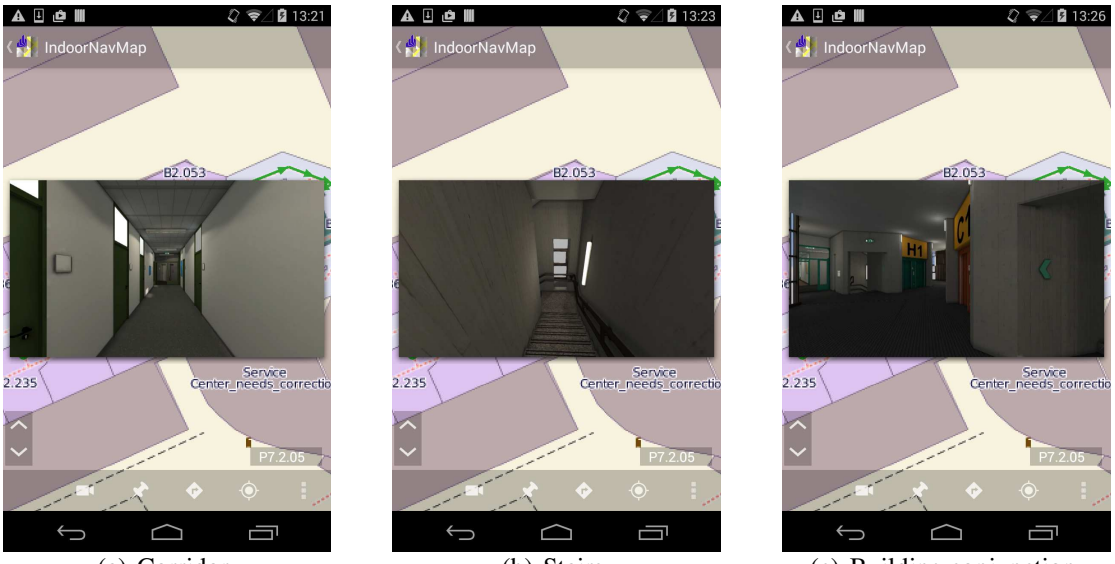
8.6. Communication Sercurity

As mentioned in section 8.1, the lighttpd server supports standardized HTTP to exchange data between the server and the clients. However, as the exchange messages in our system are very sensitive, i.e., floor plan, personal information, etc., the sercurity of the system must be taken into consideration. Fortunately, with the lighttpd server, it is possible to enable the Secure Sockets Layer (SSL) in addition to standard HTTP which allows us to use Hyper Text Transfer Protocol Secure (HTTPS) for the communication over the network. With HTTPS, all the exhange messages between the server and the clients in our system are encrypted to prevent any attack from a third party.



(a) Register (b) Login (c) After Login

Figure 8.18.: Peer group finding



(a) Corridor (b) Stairs (c) Building conjunction

Figure 8.19.: 3-D representation of the routing path

9. Conclusions

Within this thesis, the techniques to improve the accuracy of WiFi fingerprinting based indoor positioning are presented.

The accuracy of indoor positioning employing WLAN information can be enhanced by a statistical approach which is able to account for the variation of the measured RSSIs in the indoor environment. The positioning accuracy depends on two factors: training models and classification rule. As our careful study on WiFi data in indoor environment showed, RSSI measurements suffer from two problems namely censoring and dropping. As discussed in chapter 2 and chapter 4, these two problems of the indoor WiFi data have not been addressed in any previous research. Therefore, within this work, methods for estimating the parameters of the training model and classification in the presence of censored and dropped data were proposed. For parameter estimation, an EM algorithm was proposed in chapter 4 which efficiently copes with the censoring and dropping problem. The proposed EM algorithm for censored Gaussian data was proved to be a virtually biasfree and efficient estimator. Improvements in positioning accuracy are demonstrated both on artificially generated data and in real field data experiments compared to some other approaches. The experiments presented in chapter 7 showed the superiority of the statistical approach compared to a deterministic approach. It is noted that the computational demand for positioning using a parametric statistical approach is less than the other mentioned approaches.

Another problem that has been considered in this work is the mismatch between the measured data of the training device and the test devices as discussed in chapter 5 which leads to a serious reduction of the positioning accuracy. In the literature, we found only one solution which tried to address this problem using Least Squares approach where the authors assumed a linear relationship between the RSSI readings of different devices. However, what we observed is that this relation is not linear which renders the LS approach inappropriate. An effective method to cope with this problem must be developed to make WiFi signal based indoor positioning realistic. Therefore, we proposed a method for aligning the training model with the properties of the test devices while relaxing the linearity assumption in chapter 5. The proposed aligning method called "smartphone adaptation" was developed within the MLLR framework which is a very well known and successful technique for speaker adaptation in automatic speech recognition. It has to be noted that the proposed adaptation method is able to cope with censored and dropped data in the adaptation data, resulting in reliable adapted models. During the adaptation procedure, the dropping rate of the adaptation data is also estimated which will be used in the classification procedure. Experimental results presented in chapter 7 demonstrated the effectiveness of doing adaptation in indoor positioning. Assuming the RSSI readings from different devices follow one linear relationship, applying the proposed approach showed a big improvement in positioning accuracy, however, still well far below the achievable accuracy. Employing clustering approach before

doing adaptation relaxed the linearity assumption resulting in assuming piecewise linearity. As a result, better position accuracy was obtained.

Improvements in positioning accuracy can be obtained by fusing the information from different sources such as WiFi information and inertial sensor information. These two kinds of information are obtainable on most modern smartphones. Inertial navigation which utilizes the data from the built-in sensors of the smartphones is able to produce precise position estimation in a short term (time and distance) only. Unfortunately the precise positioning results cannot be maintained for a longer period of time due to error accumulation over time and distance, resulting in unreliable position estimates. WiFi fingerprinting based positioning, on the other hand, can provide a stable positioning accuracy without the error accumulation problem. As discussed in chapter 2, these two positioning techniques can be combined in order to produce better positioning results compared to using any individual approach alone. Therefore, chapter 6 presented a modified HMM for data fusion of WiFi data and inertial sensor data and utilizing the possible walking path of the user to come up with position estimates. More accurate positioning results were obtained by employing HMM in comparison with using WiFi information or inertial sensor information alone. Furthermore, a method to reduce the quantization error caused by the coarse grid of trained positions was proposed in chapter 6. By introducing pseudo states to the HMM inbetween the regular states and synthesizing the emission PDFs of the pseudo states from those of neighboring regular states, we obtained a dense grid of states without additional training effort. Experimental results showed that employing the extended HMM improved the positioning accuracy.

In addition to the theoretical research, a real server based indoor positioning and navigation system was developed as presented in chapter 8. At the moment the system is able to provide the localization and navigation services for the students of the University of Paderborn. The system employs the lighttpd web server and the FastCGI protocol which allows to handle thousands of connections in parallel on the server. In addition, the FastCGI modules, which are written in C++, run in seperate processes which make the system stable, and further, easy to scale and easy to maintain. A Java application for Android smartphones was developed which is able to run as a standalone positioning system or as a client in the system. As discussed in chapter 8, developing two possible operation modes for the smartphone application solves the problem of loss of internet connection, unsynchronization of map data and fingerprinting training model, and so on. As a result, the system seems to satisfy the requirements of low cost, high robustness and simplicity in deployment, scaling and maintenance.

Outlook

An interesting direction for future research is how to improve/update the radio map with the measured data reported by the users during the online phase. To do that, a possible solution is to develop a semi-supervised online learning system where the already built database is continuously updated during the system operation using the measurements reported by users. This kind of system can automatically handle the changes of the indoor environment or WLAN infrastructure without re-training the database from scratch after a certain amount of time. This also helps to improve the accuracy of the training model since more training data are available. The challenge is how to determine the position where the measurement is

taken. The REDPIN system believes in the information that is reported from any users which does not seem to be fault tolerant since the database can be corrupted on purpose or unintentionally through wrong user positions. Simultaneous Localization And Mapping (SLAM) is a common approach in the robotics community which mainly relies on inertial sensor information to track the position of a mobile robot and build the map simultaneously. This approach can be used to determine the position at which a specific RSSI measurement is collected. However, as discussed before, this technique suffers from the error accumulation over time and distance and, as a consequence, estimated positions are not reliable. Map matching can be formulated as a fingerprinting based localization problem where the user position is assigned to a route segment based on the online observation and training data. Therefore, the combination of SLAM and map matching approach would be a relevent solution to estimate the position of the user where RSSI measurement is collected since the accumulated error in inertial navigation are corrected globally by RSSI data and floor plan information.

Our proposed EM algorithms presented in chapter 4 are efficient for estimating the parameters of censored and dropped Gaussian data. However, we employed the empirical fixed clipping threshold for data measured by all devices. This might degrade the parameter estimation performance if the clipping threshold of different devices are not identical. Therefore, estimating clipping threshold from training data should be done before parameter estimation procedure to ensure the precision of the estimated parameters. As a result, a method for clipping threshold estimation is needed.

Employing a Gaussian mixture model estimation to estimate the RSSI distribution instead of the assumption of a single Gaussian would be an interesting try. This method might help to improve the precision of signal strength distribution estimation, since it is able to capture all the modes in the training data distribution. However, since censoring and dropping are severe in WiFi data, methods for dealing with censored and dropped data during the parameter estimation procedure would need to be taken into consideration.

A. Appendix

A.1. Derivation of EM Algorithm

A.1.1. Computation of I_0

$$\begin{aligned} I_0(\theta^{(\kappa)}) &= \int_{-\infty}^{c} \mathcal{N}\left(y;\theta^{(\kappa)}\right) \mathrm{d}y \\ &= \int_{-\infty}^{c} \frac{1}{\sqrt{2\pi}\sigma^{(\kappa)}} \exp\left(-\frac{(y-\mu^{(\kappa)})^2}{2(\sigma^2)^{(\kappa)}}\right) \mathrm{d}y \end{aligned} \tag{A.1}$$

Let $t = \frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}$ and change the variable of the integral, we arrive at:

$$I_0(\theta^{(\kappa)}) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}} \exp(-t^2)\mathrm{d}t \tag{A.2}$$

Since $\exp(-t^2)$ is an even function, the limits of the integral in Eq. (A.2) can be modified by changing their signs and swapping lower and upper limits, then $I_0(\theta^{(\kappa)})$ can be easily obtained by using complementary error function

$$I_0(\theta^{(\kappa)}) = \frac{1}{2}\mathrm{erfc}\left(-\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right) \tag{A.3}$$

A.1.2. Computation of I_1

$$\begin{aligned} I_1(\theta^{(\kappa)}) &= \int_{-\infty}^{c} y\mathcal{N}\left(y;\theta^{(\kappa)}\right) \mathrm{d}y \\ &= \int_{-\infty}^{c} y\frac{1}{\sqrt{2\pi}\sigma^{(\kappa)}} \exp\left(-\frac{(y-\mu^{(\kappa)})^2}{2(\sigma^2)^{(\kappa)}}\right) \mathrm{d}y \end{aligned} \tag{A.4}$$

Employing the integration by parts rule, let

$$\begin{cases} u = y \\ \mathrm{d}v = \frac{1}{\sqrt{2\pi}\sigma^{(\kappa)}} \exp\left(-\frac{(y-\mu^{(\kappa)})^2}{2(\sigma^2)^{(\kappa)}}\right) \mathrm{d}y \end{cases} \Rightarrow \begin{cases} \mathrm{d}u = \mathrm{d}y \\ v = \frac{1}{2}\mathrm{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right) \end{cases}, \tag{A.5}$$

then we arrive at

$$I_1(\theta^{(\kappa)}) = y\frac{1}{2}\mathrm{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right)\Bigg|_{-\infty}^{c} - \frac{1}{2}\underbrace{\int_{-\infty}^{c} \mathrm{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right) \mathrm{d}y}_{=A_1} \tag{A.6}$$

Again integration by parts rule is used for computing the remaining integal A_1 as follows, to simplify the integral, let $t = \frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}$, A_1 can be written as

$$A_1 = \sqrt{2}\sigma^{(\kappa)} \int_{-\infty}^{\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}} \text{erf}\,(t)\,\mathrm{d}t \tag{A.7}$$

Let

$$\begin{cases} u = \text{erf}\,(t) \\ \mathrm{d}v = \mathrm{d}t \end{cases} \Rightarrow \begin{cases} \mathrm{d}u = \frac{2}{\sqrt{\pi}} \exp\left(-t^2\right) \mathrm{d}t \\ v = t \end{cases}, \tag{A.8}$$

then we arrive at

$$\begin{aligned} A_1 &= \sqrt{2}\sigma^{(\kappa)} \left(t\text{erf}\,(t) \Big|_{-\infty}^{\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}} - \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}} 2t \exp\left(-t^2\right) \mathrm{d}t \right) \\ &= \sqrt{2}\sigma^{(\kappa)} \left(t\text{erf}\,(t) + \frac{1}{\sqrt{\pi}} \exp\left(-t^2\right) \right) \Big|_{-\infty}^{\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}} \end{aligned} \tag{A.9}$$

Using A_1 in Eq. (A.6), $I_1(\theta^{(\kappa)})$ is readily obtained:

$$I_1(\theta^{(\kappa)}) = \mu^{(\kappa)} I_0(\theta^{(\kappa)}) - \frac{1}{\sqrt{2\pi}} \sigma^{(\kappa)} \exp\left(- \left(\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right)^2 \right) \tag{A.10}$$

A.1.3. Computation of I_2

$$\begin{aligned} I_2(\theta^{(\kappa)}) &= \int_{-\infty}^{c} y^2 \mathcal{N}\left(y; \theta^{(\kappa)}\right) \mathrm{d}y \\ &= \int_{-\infty}^{c} y^2 \frac{1}{\sqrt{2\pi}\sigma^{(\kappa)}} \exp\left(-\frac{(y-\mu^{(\kappa)})^2}{2(\sigma^2)^{(\kappa)}} \right) \mathrm{d}y \end{aligned} \tag{A.11}$$

Employing the integration by parts rule, let

$$\begin{cases} u = y^2 \\ \mathrm{d}v = \frac{1}{\sqrt{2\pi}\sigma^{(\kappa)}} \exp\left(-\frac{(y-\mu^{(\kappa)})^2}{2(\sigma^2)^{(\kappa)}} \right) \mathrm{d}y \end{cases} \Rightarrow \begin{cases} \mathrm{d}u = 2y\mathrm{d}y \\ v = \frac{1}{2}\text{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) \end{cases}, \tag{A.12}$$

then we arrive at

$$I_2(\theta^{(\kappa)}) = y^2 \frac{1}{2}\text{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma} \right) \Big|_{-\infty}^{c} - \underbrace{\int_{-\infty}^{c} y\text{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) \mathrm{d}y}_{A_2} \tag{A.13}$$

To compute A_2, integration by parts is again applied, let

$$\begin{cases} u_1 = y \\ \mathrm{d}v_1 = \text{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) \mathrm{d}y \end{cases} \Rightarrow \begin{cases} \mathrm{d}u_1 = \mathrm{d}y \\ v_1 = \int \text{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) \mathrm{d}y \end{cases}, \tag{A.14}$$

For calculating v_1, let $t = \frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}$

$$v_1 = \sqrt{2}\sigma^{(\kappa)} \int \text{erf}\,(t)\,\mathrm{d}t \tag{A.15}$$

and applying integration by parts

$$\left\{ \begin{array}{l} u_2 = \text{erf}\,(t) \\ \mathrm{d}v_2 = \mathrm{d}t \end{array} \right. \Rightarrow \left\{ \begin{array}{l} \mathrm{d}u_2 = \frac{2}{\sqrt{\pi}} \exp\left(-t^2\right) \mathrm{d}t \\ v_2 = t \end{array} \right. , \tag{A.16}$$

we arrive at

$$\begin{aligned} v_1 &= \sqrt{2}\sigma^{(\kappa)} \left(t.\text{erf}\,(t) - \frac{1}{\sqrt{\pi}} \int 2t \exp\left(-t^2\right) \mathrm{d}t \right) \\ &= \sqrt{2}\sigma^{(\kappa)} \left(t.\text{erf}\,(t) + \frac{1}{\sqrt{\pi}} \exp\left(-t^2\right) \right). \end{aligned} \tag{A.17}$$

Using $t = \frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}$ in v_1 then we obtain

$$\begin{aligned} v_1 &= \sqrt{2}\sigma^{(\kappa)} \left\{ \frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \text{erf} \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) + \frac{1}{\sqrt{\pi}} \exp\left(-(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}})^2 \right) \right\} \\ &= (y-\mu^{(\kappa)}) \text{erf} \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) + \sigma^{(\kappa)} \sqrt{\frac{2}{\pi}} \exp\left(- \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right)^2 \right). \end{aligned} \tag{A.18}$$

Now, A_2 can be calculated as follows

$$\begin{aligned} A_2 &= u_1 v_1 \Big|_{-\infty}^{c} - \int_{-\infty}^{c} v_1 du_1 \\ &= y \left\{ (y-\mu^{(\kappa)}) \text{erf} \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) + \sigma^{(\kappa)} \sqrt{\frac{2}{\pi}} \exp\left(- \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right)^2 \right) \right\} \Big|_{-\infty}^{c} \\ &\quad - \int_{-\infty}^{c} \left\{ (y-\mu^{(\kappa)}) \text{erf} \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) + \sigma^{(\kappa)} \sqrt{\frac{2}{\pi}} \exp\left(- \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right)^2 \right) \right\} \mathrm{d}y \\ &= y \left\{ (y-\mu^{(\kappa)}) \text{erf} \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) + \sigma^{(\kappa)} \sqrt{\frac{2}{\pi}} \exp\left(- \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right)^2 \right) \right\} \Big|_{-\infty}^{c} \\ &\quad \underbrace{- \int_{-\infty}^{c} y.\text{erf} \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) \mathrm{d}y}_{A_2} + \mu^{(\kappa)} \underbrace{\int_{-\infty}^{c} \text{erf} \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right) \mathrm{d}y}_{v_1 \big|_{-\infty}^{c}} \\ &\quad - 2\left(\sigma^2\right)^{(\kappa)} \underbrace{\int_{-\infty}^{c} \frac{1}{\sqrt{2\pi}\sigma^{(\kappa)}} \exp\left(- \left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}} \right)^2 \right) \mathrm{d}y}_{I_0(\theta^{(\kappa)})} \end{aligned} \tag{A.19}$$

Simply using the limits, A_2 is readily obtained:

$$\begin{aligned} A_2 &= \frac{1}{2}y^2\text{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right)\Bigg|_{-\infty}^{c} - \frac{1}{2}\left\{(\mu^{(\kappa)2} + \left(\sigma^2\right)^{(\kappa)}).\left(\text{erf}\left(\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right)+1\right)\right. \\ &\quad \left. - \left(c+\mu^{(\kappa)}\right)\sigma^{(\kappa)}\sqrt{\frac{2}{\pi}}\exp\left(-\left(\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right)^2\right)\right\} \\ &= \frac{1}{2}y^2\text{erf}\left(\frac{y-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right)\Bigg|_{-\infty}^{c} - (\mu^{(\kappa)2} + \left(\sigma^2\right)^{(\kappa)})I_0(\theta^{(\kappa)}) \\ &\quad + \left(c+\mu^{(\kappa)}\right).\sigma^{(\kappa)}\frac{1}{\sqrt{2\pi}}\exp\left(-\left(\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right)^2\right) \end{aligned} \tag{A.20}$$

Plugging A_2 into Eq. (A.13), the calculation of I_2 ends up with

$$I_2(\theta^{(\kappa)}) = \left(\sigma^{2(\kappa)} + \mu^{2(\kappa)}\right)I_0(\theta^{(\kappa)}) - \frac{1}{\sqrt{2\pi}}\sigma^{(\kappa)}(\mu^{(\kappa)}+c)\exp\left(-\left(\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right)^2\right) \tag{A.21}$$

A.1.4. Computation of W Entries

$$\begin{aligned} \frac{\partial}{\partial\mu}I_0(\theta) &= \frac{\partial}{\partial\mu}\int_{-\infty}^{c}p(y;\theta)\text{d}y \\ &= \frac{1}{\sigma^2}\int_{-\infty}^{c}(y-\mu)p(y;\theta)\text{d}y \\ &= \frac{1}{\sigma^2}(I_1(\theta)-\mu I_0(\theta)) \end{aligned} \tag{A.22}$$

$$\begin{aligned} \frac{\partial}{\partial\mu}I_1(\theta) &= \frac{\partial}{\partial\mu}\int_{-\infty}^{c}y.p(y;\theta)\text{d}y \\ &= \frac{1}{\sigma^2}\int_{-\infty}^{c}y(y-\mu)p(y;\theta)\text{d}y \\ &= \frac{1}{\sigma^2}(I_2(\theta)-\mu I_1(\theta)) \end{aligned} \tag{A.23}$$

$$\begin{aligned} \frac{\partial}{\partial\mu}I_2(\theta) &= \frac{\partial}{\partial\mu}\int_{-\infty}^{c}y^2.p(y;\theta)\text{d}y \\ &= \frac{1}{\sigma^2}\int_{-\infty}^{c}y^2(y-\mu)p(y;\theta)\text{d}y \\ &= \frac{1}{\sigma^2}(I_3(\theta)-\mu I_2(\theta)) \end{aligned} \tag{A.24}$$

$$\begin{aligned} \frac{\partial}{\partial\sigma^2}I_0(\theta) &= \frac{\partial}{\partial\sigma^2}\int_{-\infty}^{c}p(y;\theta)\text{d}y \\ &= -\frac{1}{2\sigma^2}I_0(\theta) + \frac{1}{2\sigma^4}\left(I_2(\theta) - 2I_1(\theta)\mu + I_0(\theta)\mu^2\right) \end{aligned} \tag{A.25}$$

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}I_1(\theta) &= \frac{\partial}{\partial\sigma^2}\int_{-\infty}^{c} yp(y;\theta)\mathrm{d}y \\ &= -\frac{1}{2\sigma^2}I_1(\theta) + \frac{1}{2\sigma^4}\left(I_3(\theta) - 2I_2(\theta)\mu + I_1(\theta)\mu^2\right) \end{aligned} \tag{A.26}$$

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}I_2(\theta) &= \frac{\partial}{\partial\sigma^2}\int_{-\infty}^{c} y^2p(y;\theta)\mathrm{d}y \\ &= -\frac{1}{2\sigma^2}I_2(\theta) + \frac{1}{2\sigma^4}\left(I_4(\theta) - 2I_3(\theta)\mu + I_2(\theta)\mu^2\right) \end{aligned} \tag{A.27}$$

where $I_3(\theta)$ and $I_4(\theta)$ can be computed by using integration by parts as the computation of $I_2(\theta)$, however lengthier computation

$$\begin{aligned} I_3(\theta) &= \int_{-\infty}^{c} y^3p(y;\theta)\mathrm{d}y \\ &= \mu\left(3\sigma^2+\mu^2\right)I_0(\theta) - \frac{1}{\sqrt{2\pi}}\sigma[2\sigma^2+\mu^2+\mu c+c^2]\exp\left(-\left(\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right)^2\right) \end{aligned} \tag{A.28}$$

$$\begin{aligned} I_4(\theta) &= \int_{-\infty}^{c} y^4p(y;\theta)\mathrm{d}y \\ &= (3\sigma^4+6\mu^2\sigma^2+\mu^4)I_0(\theta) \\ &\quad - \frac{1}{\sqrt{2\pi}}\sigma[\sigma^2(5\mu+3c)+\mu^3+\mu^2c+\mu c^2+c^3]\exp\left(-\left(\frac{c-\mu^{(\kappa)}}{\sqrt{2}\sigma^{(\kappa)}}\right)^2\right) \end{aligned} \tag{A.29}$$

A.1.5. Computation of Information Matrix I

For convenience, the log-likelihood function and the information matrix are recalled as follows

$$\begin{aligned}\ln p(x;\theta) = \ln\left(\frac{N!}{M!(N-M)!}\right) + (N-M)\ln\left(I_0(\theta)\right) \\ - \frac{M}{2}\ln\left(2\pi\sigma^2\right) - \frac{1}{2}\sum_{j=1}^{M}\left(\frac{x_j-\mu}{\sigma}\right)^2. \end{aligned} \tag{A.30}$$

$$\mathbf{I} = \begin{pmatrix} \mathrm{E}\left[-\frac{\partial^2}{\partial\mu^2}\ln p(\mathbf{x};\theta)\right] & \mathrm{E}\left[-\frac{\partial^2}{\partial\mu\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] \\ \mathrm{E}\left[-\frac{\partial^2}{\partial\mu\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] & \mathrm{E}\left[-\frac{\partial^2}{\partial\sigma^2\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] \end{pmatrix}. \tag{A.31}$$

In the following, to shorten the notation, the parameters of the I_j functions are removed. Here the derivative computations of the I_j which were given in Eq. (A.22) to (A.27) are employed. Since I_0 is the probability that a measurement is censored, in the following derivation, the expected number of uncensored measurements is determined by $\mathrm{E}[M] = N(1-I_0)$, where N is the total number of measurements.

- $\mathrm{E}\left[-\frac{\partial^2}{\partial\mu^2}\ln p(\mathbf{x};\theta)\right]$:
  First derivative of log-likelihood function w.r.t. μ

$$\begin{aligned}\frac{\partial}{\partial\mu}\ln p(\mathbf{x};\theta) &= (N-M)\frac{1}{I_0}\frac{\partial}{\partial\mu}I_0 - \frac{1}{2}\sum_{j=1}^{M}\left(-2\frac{x_j-\mu}{\sigma^2}\right)\\ &= (N-M)\frac{1}{I_0}\frac{1}{\sigma^2}\big(I_1-\mu I_0\big) + \frac{1}{\sigma^2}\sum_{j=1}^{M}(x_j-\mu)\end{aligned} \tag{A.32}$$

  Second derivative of log-likelihood function w.r.t. μ:

$$\begin{aligned}\frac{\partial^2}{\partial\mu^2}\ln p(\mathbf{x};\theta) &= \frac{\partial}{\partial\mu}\left((N-M)\frac{1}{I_0}\frac{1}{\sigma^2}\big(I_1-\mu I_0\big) + \frac{1}{\sigma^2}\sum_{j=1}^{M}(x_j-\mu)\right)\\ &= \frac{N-M}{\sigma^2}\frac{\partial}{\partial\mu}\left(\frac{I_1}{I_0}-\mu\right) - \frac{M}{\sigma^2}\\ &= \frac{N-M}{\sigma^2}\left(\frac{\frac{\partial I_1}{\partial\mu}I_0 - I_1\frac{\partial I_0}{\partial\mu}}{I_0^2} - 1\right) - \frac{M}{\sigma^2}\\ &= \frac{N-M}{\sigma^2}\left(\frac{\frac{1}{\sigma^2}(I_2-\mu I_1)I_0 - I_1\frac{1}{\sigma^2}(I_1-\mu I_0)}{I_0^2} - 1\right) - \frac{M}{\sigma^2}\\ &= \frac{N-M}{\sigma^4}\left(\frac{I_2I_0-I_1^2}{I_0^2} - \sigma^2\right) - \frac{M}{\sigma^2}\\ &= \frac{N-M}{\sigma^4}\left(\frac{I_2I_0-I_1^2}{I_0^2}\right) - \frac{N}{\sigma^2}\end{aligned} \tag{A.33}$$

  Expectation of the second derivative of log-likelihood function w.r.t. μ:

$$\begin{aligned}E\left[-\frac{\partial^2}{\partial\mu^2}\ln p(\mathbf{x};\theta)\right] &= \mathrm{E}\left[-\frac{N-M}{\sigma^4}\left(\frac{I_2I_0-I_1^2}{I_0^2}\right) + \frac{N}{\sigma^2}\right]\\ &= -\mathrm{E}\left[N-M\right]\frac{1}{\sigma^4}\left(\frac{I_2I_0-I_1^2}{I_0^2}\right) + \frac{N}{\sigma^2}\\ &= -NI_0\frac{1}{\sigma^4}\left(\frac{I_2I_0-I_1^2}{I_0^2}\right) + \frac{N}{\sigma^2}\\ &= \frac{N}{\sigma^2}\left(\frac{I_1^2-I_2I_0}{I_0} + 1\right)\end{aligned} \tag{A.34}$$

- $\mathrm{E}\left[-\frac{\partial^2}{\partial\mu\partial\sigma^2}\ln p(\mathbf{x};\theta)\right]$:

$$\begin{aligned}\frac{\partial^2}{\partial\mu\partial\sigma^2}\ln p(\mathbf{x};\theta) &= \frac{\partial}{\partial\sigma^2}\left((N-M)\frac{1}{I_0}\frac{1}{\sigma^2}\big(I_1-\mu I_0\big)+\frac{1}{\sigma^2}\sum_{j=1}^{M}(x_j-\mu)\right)\\ &= (N-M)\frac{\partial}{\partial\sigma^2}\left\{\frac{1}{\sigma^2}\left(\frac{I_1}{I_0}-\mu\right)\right\}-\frac{1}{\sigma^4}\sum_{j=1}^{M}(x_j-\mu)\\ &= (N-M)\left\{-\frac{1}{\sigma^4}\left(\frac{I_1}{I_0}-\mu\right)+\frac{1}{\sigma^2}\left(\frac{\frac{\partial I_1}{\partial\sigma^2}I_0-I_1\frac{\partial I_0}{\partial\sigma^2}}{I_0^2}\right)\right\}\\ &\quad-\frac{1}{\sigma^4}\sum_{j=1}^{M}(x_j-\mu)\cdot \end{aligned} \tag{A.35}$$

Expectation of the second derivative of log-likelihood function w.r.t. μ and σ^2:

$$\begin{aligned}\mathrm{E}\left[-\frac{\partial^2}{\partial\mu\partial\sigma}\ln p(\mathbf{x};\theta)\right] &= NI_0\left\{\frac{1}{\sigma^4}\left(\frac{I_1}{I_0}-\mu\right)-\frac{1}{\sigma^2}\left(\frac{\frac{\partial I_1}{\partial\sigma^2}I_0-I_1\frac{\partial I_0}{\partial\sigma^2}}{I_0^2}\right)\right\}\\ &\quad+\frac{1}{\sigma^4}\mathrm{E}\left[\sum_{j=1}^{M}(x_j-\mu)\right],\end{aligned} \tag{A.36}$$

where the remaining expectation can be computed as follows

$$\begin{aligned}\mathrm{E}\left[\sum_{j=1}^{M}(x_j-\mu)\right] &= \mathrm{E}\left[M\right]\left(\mathrm{E}\left[x\right]-\mu\right)\\ &= N(1-I_0)\left(\frac{1}{1-I_0}\int_c^{\infty} yp(y;\theta)dy-\mu\right)\\ &= N(1-I_0)\left(\frac{1}{1-I_0}(\mu-I_1)-\mu\right)\\ &= N(\mu I_0-I_1).\end{aligned} \tag{A.37}$$

Using this in Eq. (A.36) we arrive at

$$E\left[-\frac{\partial^2}{\partial\mu\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] = \frac{N}{\sigma^2}\left(-\frac{\frac{\partial I_1}{\partial\sigma^2}I_0-I_1\frac{\partial I_0}{\partial\sigma^2}}{I_0}\right) \tag{A.38}$$

- $E\left[-\frac{\partial^2}{\partial\sigma^2\partial\sigma^2}\ln p(\mathbf{x};\theta)\right]$:

First derivative of log-likelihood function w.r.t. σ^2

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}\ln p(\mathbf{x};\theta) &= (N-M)\frac{1}{I_0}\frac{\partial}{\partial\sigma^2}I_0-\frac{M}{2}\frac{1}{\sigma^2}-\frac{1}{2}\left(-\frac{1}{\sigma^4}\right)\sum_{j=1}^{M}(x_j-\mu)^2\\ &= (N-M)\left\{-\frac{1}{2\sigma^2}+\frac{1}{2\sigma^4}\left(\frac{I_2}{I_0}-2\mu\frac{I_1}{I_0}+\mu^2\right)\right\}\\ &\quad-\frac{M}{2\sigma^2}+\frac{1}{2\sigma^4}\sum_{j=1}^{M}(x_j-\mu)^2\end{aligned} \tag{A.39}$$

Second derivative of log-likelihood function w.r.t. σ^2

$$\begin{aligned}\frac{\partial^2}{\partial\sigma^2\partial\sigma^2}\ln p(\mathbf{x};\theta) &= (N-M)\left\{\frac{1}{2\sigma^4}-\frac{1}{\sigma^6}\left(\frac{I_2}{I_0}-2\mu\frac{I_1}{I_0}+\mu^2\right)\right.\\ &\quad\left.+\frac{1}{2\sigma^4}\frac{\partial}{\partial\sigma^2}\left(\frac{I_2}{I_0}-2\mu\frac{I_1}{I_0}+\mu^2\right)\right\}+\frac{M}{2\sigma^4}-\frac{1}{\sigma^6}\sum_{j=1}^{M}(x_j-\mu)^2\\ &= (N-M)\left\{\frac{1}{2\sigma^4}-\frac{1}{\sigma^6}\left(\frac{I_2}{I_0}-2\mu\frac{I_1}{I_0}+\mu^2\right)\right.\\ &\quad\left.+\frac{1}{2\sigma^4}\left(\frac{\frac{\partial I_2}{\partial\sigma^2}I_0-I_2\frac{\partial I_0}{\partial\sigma^2}}{I_0^2}-2\mu\frac{\frac{\partial I_1}{\partial\sigma^2}I_0-I_1\frac{\partial I_0}{\partial\sigma^2}}{I_0^2}\right)\right\}\\ &\quad+\frac{M}{2\sigma^4}-\frac{1}{\sigma^6}\sum_{j=1}^{M}(x_j-\mu)^2 \qquad \text{(A.40)}\end{aligned}$$

Expectation of the second derivative of log-likelihood function w.r.t. σ^2:

$$\begin{aligned}\mathrm{E}\left[-\frac{\partial^2}{\partial\sigma^2\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] &= -NI_0\left\{\frac{1}{2\sigma^4}-\frac{1}{\sigma^6}\left(\frac{I_2}{I_0}-2\mu\frac{I_1}{I_0}+\mu^2\right)\right.\\ &\quad\left.+\frac{1}{2\sigma^4}\left(\frac{\frac{\partial I_2}{\partial\sigma^2}I_0-I_2\frac{\partial I_0}{\partial\sigma^2}}{I_0^2}-2\mu\frac{\frac{\partial I_1}{\partial\sigma^2}I_0-I_1\frac{\partial I_0}{\partial\sigma^2}}{I_0^2}\right)\right\}\\ &\quad-N(1-I_0)\frac{1}{2\sigma^4}+\frac{1}{\sigma^6}\mathrm{E}\left[\sum_{j=1}^{M}(x_j-\mu)^2\right] \qquad \text{(A.41)}\end{aligned}$$

where

$$\begin{aligned}\mathrm{E}\left[\sum_{j=1}^{M}(x_j-\mu)^2\right] &= \mathrm{E}\left[M\right]\mathrm{E}\left[(x-\mu)^2\right]\\ &= N(1-I_0)\left(\frac{1}{1-I_0}\int_c^{\infty}(y-\mu)^2p(y;\theta)dy\right)\\ &= N\left(\sigma^2-I_2+2\mu I_1-\mu^2I_0\right) \qquad \text{(A.42)}\end{aligned}$$

Using this in Eq. (A.41) we arrive at

$$\mathrm{E}\left[-\frac{\partial^2}{\partial\sigma^2\partial\sigma^2}\ln p(\mathbf{x};\theta)\right] = \frac{N}{\sigma^2}\left\{\frac{1}{2\sigma^2}-\frac{1}{2\sigma^2}\left(\frac{\frac{\partial I_2}{\partial\sigma^2}I_0-I_2\frac{\partial I_0}{\partial\sigma^2}}{I_0}-2\mu\frac{\frac{\partial I_1}{\partial\sigma^2}I_0-I_1\frac{\partial I_0}{\partial\sigma^2}}{I_0}\right)\right\} \qquad \text{(A.43)}$$

A.2. Parameters of Kalman Filter

Time interval length $T = 20\,\mathrm{ms}$.

$$\mathbf{F} = \begin{pmatrix} 0.999 & 0.01 \\ 0 & 1 \end{pmatrix} \tag{A.44}$$

$$\mathbf{G} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{A.45}$$

$$\mathbf{H} = \begin{pmatrix} 0.99 & 0.01 \\ 0 & 1 \end{pmatrix} \tag{A.46}$$

$$\mathbf{Q}_1 = \begin{pmatrix} 0.00001 & 0 \\ 0 & 0.0006 \end{pmatrix} \tag{A.47}$$

$$\mathbf{Q}_2 = \begin{pmatrix} 0.00006 & 0 \\ 0 & 0.0000001 \end{pmatrix} \tag{A.48}$$

A.3. Android Smartphone Application

A.3.1. Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="de.nt.INM"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="17"  android:targetSdkVersion="19"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
        <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
        <uses-permission android:name="android.permission.READ_PHONE_STATE" />
        <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
        <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
        <uses-permission android:name="android.permission.INTERNET"/>
        <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
        <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
        <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>

    <application android:name="de.nt.INM.Main.MapScrollApplication"
        android:uiOptions="splitActionBarWhenNarrow"
        android:allowBackup="true"
        android:largeHeap="true"
        android:icon="@drawable/logo"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity android:name="de.nt.INM.Main.IndoorNavMap"
                  android:label="@string/app_name">
            <intent-filter>

                <action android:name="android.net.wifi.WIFI_STATE_CHANGED"/>
                <action android:name="android.net.wifi.SCAN_RESULTS"/>

                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="de.nt.INM.Util.SettingsActivity"
                  android:label="Settings">
        </activity>
```

```
        <activity android:name="de.nt.INM.Social.Account.AccountSettingsActivity"
                  android:label="Account Settings">
        </activity>
        <activity android:name="de.nt.INM.Social.Friends.FriendsActivity"
                  android:label="Friends">
        </activity>
        <activity android:name="de.nt.INM.Social.LearningGroups.LearningGroupsActivity"
                  android:label="Learning Groups">
        </activity>
        <activity android:name="de.nt.INM.Social.LearningGroups.
            LearningGroupsListActivity"
                  android:label="Learning Groups">
        </activity>
        <activity android:name="de.nt.INM.Social.Account.
            AccountSettingsChangeAvatarActivity"
                  android:label="Change Avatar">
        </activity>
        <activity android:name="de.nt.INM.Measurement.WLANDataAssemblerActivity"
                  android:label="WDA"
                  android:screenOrientation="portrait">
        </activity>
         <activity android:name="de.nt.INM.GUI.StreamActivity"
                  android:label="Video Guide">
        </activity>

        <service android:name="de.nt.INM.Util.MapDownloadService" ></service>
        <service android:name="de.nt.INM.Localization.LocalizationService" ></service>
    </application>
</manifest>
```

List A.1: Manifest file of the smartphone application

A.4. Requests and Responses of the Communication in Indoor Navigation System

A.4.1. Set FringerPrint Request and Response

List A.2 shows an example of a SetFingerprintRequest which is sent by client.

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<SetFingerprintRequests>
  <Smartphone HardwareType="Nexus 4" HardwareID="355136050665226" />
  <SetFingerprintRequest Date="2013-07-22 08:33:34">
    <Marker Floor="0" X="8.7678" Y="51.7088" Orientation="280.47" />
    <AP MAC="b4:e9:b0:32:c3:b1" strength="-55" />
    <AP MAC="b4:e9:b0:32:c3:b0" strength="-56" />
    <AP MAC="b4:e9:b0:32:c3:b3" strength="-57" />
    <AP MAC="b4:e9:b0:32:c3:bf" strength="-68" />
    <AP MAC="b4:e9:b0:32:c3:be" strength="-68" />
    <AP MAC="b4:e9:b0:32:c3:bc" strength="-69" />
    <AP MAC="00:26:cb:93:be:41" strength="-81" />
    <AP MAC="a4:0c:c3:d1:db:50" strength="-83" />
    <PressureSensor>997.7616</PressureSensor>
  </SetFingerprintRequest>
</SetFingerprintRequests>
```

List A.2: Set Fingerprint Request

List A.3 shows an example of a SetFingerprintRequestAnswer which is generated by server

for the case of successfully adding new measurements to the database.

```
<?xml version='1.0' standalone='yes' ?>
<SetFingerprintRequestAnswer error="0" description="OK">
</SetFingerprintRequestAnswer>
```

List A.3: Set Fingerprint Response

List of abbreviations

AP	***A**ccess **P**oint*
AOA	***A**ngle **O**f **A**rrival*
BLE	***B**luetooth **L**ow **E**nergy*
CDF	***C**umulative **D**istribution **F**unction*
CGI	***C**ommon **G**ateway **I**nterface*
CRLB	***C**ramer-**R**ao **L**ower **B**ound*
DOA	***D**irection **O**f **A**rrival*
DR	***D**ead **R**eckoning*
EM	***E**xpectation-**M**aximization*
GIS	***G**eographic **I**nformation **S**ystem*
GLONASS	***GLO**bal **NA**vigation **S**atellite **S**ystem*
GMM	***G**aussian **M**ixture **M**odel*
GPS	***G**lobal **P**ositioning **S**ystem*
GSM	***G**lobal **S**ystem for **M**obile communications*
HMM	***H**idden **M**arkov **M**odel*
HTTP	***H**yper **T**ext **T**ransfer **P**rotocol*
HTTPS	***H**yper **T**ext **T**ransfer **P**rotocol **S**ecure*
ID	***Id**entification*
IMU	***I**ertial **M**easurement **U**nit*
IPS	***I**ndoor **P**ositioning **S**ystem*
JOSM	***J**ava **O**pen**S**treetMap editor*
LBS	***L**ocation-**B**ased **S**ervice*
LOS	***L**ine **O**f **S**ight*
LS	***L**east **S**quares*
MAC	***M**edia **A**ccess **C**ontrol*
MEMS	***M**icro**e**lectro**m**echanical **S**ystems*
ML	***M**aximum **L**ikelihood*
MLLR	***M**aximum **L**ikelihood **L**inear **R**egression*
MSE	***M**ean **S**quare **E**rror*
NLOS	***N**one **L**ine **O**f **S**ight*
NIC	***N**etwork **I**nterface **C**ontroller*
OSM	***O**pen **S**treet **M**ap*
PDF	***P**robability **D**ensity **F**unction*
RSSI	***R**eceived **S**ignal **S**trength **I**ndex*
SQL	***S**tructure **Q**uery **L**anguage*
SSL	***S**ecure **S**ockets **L**ayer*
TDOA	***T**ime **D**ifference **O**f **A**rrival*

TLS	***T**otal **L**east **S**quares*
TOA	***T**ime **O**f **A**rrival*
TOF	***T**ime **O**f **F**light*
US	***U**nited **S**tates*
WLAN	***W**ireless **L**ocal **A**rea **N**etwork*
WLS	***W**eighted **L**east **S**quares*
w.r.t.	***w**ith **r**espect **t**o*
WTLS	***W**eighted **T**otal **L**east **S**quares*
XML	***Ex**tensible **M**arkup **L**anguage*

Notations and Symbols

General Notations and Functions

$\mathrm{E}[\cdot]$	Expectation
$(\cdot)^T$	Transpose
$Q(\cdot)$	Auxiliary function
$(\cdot)^{-1}$	Inversion
$\ln(\cdot)$	Natural logarithm function
$p(\mathbf{x}\|\ell_k)$	Class conditional probability density function
$\delta(\cdot)$	Diract delta function
$\mathrm{erf}(\cdot)$	Error function
$\mathrm{erfc}(\cdot)$	Complementary error function
$\mathcal{N}(\theta)$	Gaussian distribution parameterized by θ
$p(\mathbf{x};\theta)$	Probability density function parameterized by θ
$\sum$	Summation of a sequence
$\prod$	Products of a sequence
$(.)!$	Factorial of a non negative integer

Fundamentals of Indoor Positioning and State of Research

$\mathcal{R}$	Mathematical expression of radio map in fingerprinting based techniques
ℓ_k	Vector consists of coordinates of the k-th position
$\mathcal{F}_k$	Fingerprint at the k-th position
$\mathcal{X}_k$	Set of measurement vectors at the k-th position
$\mathbf{x}_{k,n}$	The n-th measurement vector at the k-th position
$\mathcal{D}(\mathbf{o},\mathbf{x}_{k,n})$	Euclidian distance between online sample $\mathbf{o}$ and training sample $\mathbf{x}_{k,n}$
θ_k	Set of the parameters of the Gaussian describing the signal strength distribution at the k-th position
$\boldsymbol{\mu}_k$	Mean vector of the Gaussian describing the signal strength distribution at the k-th position
$\boldsymbol{\Sigma}_k$	Covariance matrix of the Gaussian describing the signal strength distribution at the k-th position

Parameter Estimation of Censored and Dropped Gaussian Data

c	Clipping threshold
Θ	Set of parameters of a GMM
π_k	Mixing weight of the k-th component of a GMM
θ_k	Set of the parameters of the k-th component of a GMM
$\mathcal{X}$	Set of observable data
$\mathcal{Z}$	Set of hidden variables
$\mathbf{y}$	Set of unobservable, non-censored, possibly dropped data
y_n	The n-th measurement, scalar value
$\mathbf{x}$	Set of observable data
θ	Set of the parameters of a univariate single Gaussian
μ	Mean of a univariate single Gaussian
σ	Variance of a univariate single Gaussian
$\theta^{(\kappa)}$	Set of the estimated parameters of a univariate single Gaussian after the κ-th iteration of an EM algorithm
$I_j\left(\theta^{(\kappa)}\right)$	The j-th moment of the truncated part of a censored Gaussian, defined in Eq. (4.8)
z_n	Realization of the binary random variable Z_n indicate whether the n-th measurement is censored ($z_n = 1$) or not ($z_n = 0$)
N	Number of measurements
M	Number of observable measurements
$\tilde{\mu}^{(\kappa)}$	Difference between the estimated mean after the κ-th EM iteration and the true mean
$(\tilde{\sigma}^2)^{(\kappa)}$	Difference between the estimated variance after the κ-th EM iteration and the true variance
$\mathbf{I}$	Information matrix
d_n	Hidden variables indicate whether the n-th measurement is dropped ($d_n = 1$) or not ($d_n = 0$)
π	Dropping rate, defined as $\pi = P(d_n = 1)$
$\beta_n(d, z)$	Probability that $d_n = 0$ or $d_n = 1$ given $z_n = 0$ or $z_n = 1$

Smartphone Adaptation within MLLR Framework

$\boldsymbol{\mu}_k$	Mean vector consists of means of all APs at the k-th position
$\boldsymbol{\xi}_k$	Extended mean vector consists of means of all APs at the k-th position and a offset term factor
$\hat{\boldsymbol{\mu}}_k$	Adapted mean vector at the k-th position
N_{AP}	Total number of observable APs
$c(k)$	Regression class $c(k)$ where the Gaussian describing k-th position belongs to
C	Total number of regression classes
$\mathbf{W}^{(c(k))}$	Mean transformation matrix to be applied to the regression class $c(k)$
$\gamma_k(n)$	The posterior probability that RSSI measurement vector $\mathbf{x}_n$ is from position ℓ_k

$\mathcal{Y}$	Set of measurement vectors
$\mathbf{y}_n$	The n-th measurement vector consists of RSSI from N_{AP} APs
$y_{n,i}$	The non-censored, possibly dropped measured data of the n-th measurement from the i-th AP
$\mathcal{D}$	Set of hidden variable vectors
$\mathbf{d}_n$	The n-th hidden variable vector consists of random variables, each indicates whether the measured data from corresponding AP is dropped or not
$d_{n,i}$	Random variable indicates whether the n-th measurement from the i-th AP is dropped or not
$\mathcal{X}$	Set of observable, censored, possibly dropped measurement vectors
$\mathbf{x}_n$	The n-th measurement vector consists of observable, censored, possibly dropped measured data from N_{AP} APs
$x_{n,i}$	The censored, possibly dropped measured data of the n-th measurement from the i-AP
$\mathbf{w}_i^{(c(k))}$	The row vector which is the i-th row of $\mathbf{W}^{(c(k))}$
λ	Set of parameters, i.e., dropping rate, mean and variance adaptation matrices, to be estimated for adaptation purpose
$\lambda_{k,i}$	Set of parameters, i.e., dropping rate, mean and variance adaptation matrices, to be estimated of the i-th AP at the k-th position
$\pi_{k,i}$	Dropping rate of the adaptation data of the i-th AP at the k-th position
$\mu_{k,i}$	Mean of the Gaussian describing the RSSI distribution of the i-th AP at the k-th position
$\sigma_{k,i}$	Variance of the Gaussian describing the RSSI distribution of the i-th AP at the k-th position
$z_{n,i}$	Random variable indicates whether the n-th measurement from the i-th AP is observed or not
$\beta_{k,n,i}(d,z)$	Probability that $d_{n,i}=0$ or $d_{n,i}=1$ given $z_{n,i}=0$ or $z_{n,i}=1$ at the k-th position
$\hat{\mathbf{\Sigma}}_k$	Adapted covariance matrix of the Gaussian describing the RSSI readings of the APs at the k-th position
$\mathbf{B}_k$	The Choleski factor of $\mathbf{\Sigma}_k$
$\mathbf{H}^{(c(k))}$	Variance transformation matrix to be applied to the regression class $c(k)$
$h_i^{(c(k))}$	The i-th component of the main diagonal of $\mathbf{H}^{(c(k))}$

Hidden Markov Model for Indoor User Tracking

s_t	Hidden state variable at time t
$\mathbf{x}_t$	RSSI observation at time t
$P_{i,j}$	Transition probability from the i-th to the j-th state
$\mathbf{v}_t$	Two-dimensional movement vector computed from inertial sensor data

$\alpha_t(k)$	Forward variable: The probability that the user is at time t in state k given the observed sequence of $\mathbf{o}_{1:t}$ and $\mathbf{v}_{1:t}$
$\boldsymbol{a}$	3-dimentional acceleration data vector
a_x	Acceleration data along x-axis
a_y	Acceleration data along y-axis
a_z	Acceleration data along z-axis
G	Gravity constant
$\boldsymbol{m}$	Magnetor data vector
$\boldsymbol{g}$	Gravity data vector
$\boldsymbol{gyr}$	Gyroscope data vector
R	Rotation matrix
ψ	Azimuth angle
$\boldsymbol{\alpha}(n)$	State vector that contains the absolute Azimuth angle and its derivative
$\mathbf{F}(n)$	Transition matrix which indicates a transition of the system from time n to $n+1$ in Kalman filter
$\boldsymbol{\nu}_1(n)$	White Gaussian system noise in Kalman filter
$\boldsymbol{\nu}_2(n)$	White Gaussian measurement noise in Kalman filter
$\mathbf{G}(n)$	Matrix that describes the influences of the system noise on the state vector in Kalman filter
$\mathbf{Q}_1$	Covariance matrix of the white Gaussian system noise in Kalman filter
$\mathbf{Q}_2$	Covariance matrix of the white Gaussian measurement noise in Kalman filter
$\mathbf{H}(n)$	Measurement matrix that describes the influences of the measurement on the state vector in Kalman filter
L_{step}	Step length

Experimental Results on Indoor Positioning

λ	Weighting factor between inertial sensor information and WiFi information in the calculation of forward variable

List of figures

List of tables

References

[1] R. Mautz, "Indoor Positioning Technologies," in *Habilitation Thesis submitted to ETH Zurich*, Zurich, February 2012.

[2] "Indoor Positioning and Navigation," http://www.computer.org/portal/web/computingnow/archive/june2013.

[3] A. Kushki, K. N. Plataniotis, and A. N. Venetsapopoulos, *WLAN Positioning Systems*, Cambridge, New York, United States, 2012.

[4] M. D. Rodríguez, J. Favela, E. A. Martínez, and M. A. Muñoz, "Location-aware access to hospital information and services," *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, no. 4, pp. 448–455, 2004.

[5] "GPS website," http://www.gps.gov/.

[6] P. Enge and P. Misra, "Special Issue on Global Positioning System," in *Proceedings of the IEEE*, January 1999, vol. 87, pp. 3–15.

[7] A. Varshavsky, M. Y. Chen, E. Lara, J. Froehlich, D. Haehnel, J. Hightower, A. LaMarca, F. Potter, T. Sohn, K. Tang, and I. Smith, "Are GSM phones THE solution for localization," in *Proceedings of the 7th IEEE Workshop on Mobile Computing Systems and Applications*, Orcas Island, WA, August 2006.

[8] R. Haeb-Umbach and S. Peschke, "A Novel Similarity Measure for positioning Cellular Phones by a Comparison With a Database of Signal Power Levels," *IEEE Transactions on Vehicular Technology*, pp. 368–372, 2007.

[9] M. Ibrahim and M. Youssef, "CellSense: A Probabilistic RSSI-based GSM Positioning System," in *Proceedings of the GLOBECOM*, Miami, PL, USA, December 2010.

[10] S. Peschke, M. Bevermeier, and R. Haeb-Umbach, "A GPS positioning approach exploiting GSM velocity estimates," in *Proceedings of the 6th Workshop on Positioning Navigation and Communication (WPNC 2009)*, Dresden, Germany, 2009.

[11] S. Panzieri, F. Pascucci, and G. Ulivi, "An outdoor navigation system using GPS and inertial platform," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 2, pp. 134–142, 2002.

[12] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 6, pp. 1067–1080, 2007.

[13] H. Cho, H. Jang, and Y. Baek, "Practical localization system for consumer devices using zigbee networks," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1562–1569, 2010.

[14] J. Yoon, J. Kim, W. Lee, and D. Eom, "A TDoA-Based Localization Using Precise Time-Synchronization," in *Proceedings of the 14th International Conference on Advanced Communication Technology (ICACT)*, PyeongChang, February 2012.

[15] X. Li and K. Pahlavan, "Super-resolution toa estimation with diversity for indoor geolocation," *IEEE Transactions on Wireless Communications*, vol. 3, no. 1, pp. 224–234, 2004.

[16] A. Ali and A.S. Omar, "Time of Arrival Estimation for WLAN Indoor Positioning Systems using Matrix Pencil Super Resolution Algorithm ," in *Proceedings of the 2th Workshop on Positioning, Navigation and Communication (WPNC)*, Hannover, Germany, March 2005.

[17] M. Li and Y. Lu, "Angle-Of-Arrival Estimation for Localization and Communication in Wireless Networks," in *Proceedings of the 16th European Signal Processing Conference (EUSIPCO)*, Lausanne, Switzerland, August 2008.

[18] P. Bahl and V.N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE, 2000, vol. 2, pp. 775–784.

[19] H. Nurminen, J. Talvitie, S. Ali-Löytty, P. Müller, E. Lohan, R. Piché, and M. Renfors, "Statistical path loss parameter estimation and positioning using rss measurements," *Journal of Global Positioning Systems*, vol. 12, no. 1, pp. 13–27, 2013.

[20] A. W. S. Au, C. Feng, S. Valaee, S. Reyes, S. Sorour, D. Gold, K. Gordon, and M. Eizenman, "Indoor tracking and navigation using received signal strength and compressive sensing on a mobile device," *IEEE Transactions on Mobile Computing*, vol. 12, no. 10, pp. 2050–2062, 2013.

[21] P. Bahl and V.N. Padmanabhan, "Enhancements to the RADAR User Location and Tracking System," in *Technical Report MSR-TR-2000-12*. Microsoft Research, February 2000.

[22] K. Kaemarungsi and P. Krishnamurth, "Modeling of indoor positioning systems based on location fingerprinting," in *Proceedings of the INFOCOM*, Hong Kong, March 2004.

[23] T. Roos, P. Myllymaki, H. Tirri, P. Misikangas, and J. Sievanen, "A probabilistic approach to wlan user location estimation," *International Journal of Wireless Information Networks*, vol. 9, no. 3, pp. 155–164, 2002.

[24] A. Agiwal, P. Khandpur, and H. Saran, "Locator: location estimation system for wireless LANs," in *Proceedings of the 2nd ACM International workshop on Wireless mobile applications and services on WLAN hostpots*. ACM, 2004, pp. 102–109.

[25] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proceedings of the 3rd International Conference on Mobile systems, applications, and services (MobiSys)*, Seattle, WA, June 2005.

[26] G. Lui, T. Gallagher, B. Li, A. G. Dempster, and C. Rizos, "Differences in RSSI Readings Made by Different Wi-Fi Chipsets: A Limitation of WLAN Localization," in *Proceedings of the International Conference on Localization and GNSS*, Tampere, June 2011, IEEE.

[27] Q. Ladetto and B. Merminod, "Digital magnetic compass and gyroscope integration for pedestrian navigation," in *Proceedings of the 9th Saint Petersburg International Conference on Integrated Navigation Systems*, Hannover, Germany, May 2002.

[28] S. Beauregard and H. Haas, "Pedestrian Dead Reckoning: A Basis for Personal Positioning," in *Proceedings of the 3th Workshop on Positioning Navigation and Communication (WPNC 2006)*, Hannover, Germany, March 2006.

[29] J. Seitz and T. Vaupel and S. Meyer and J. G. Boronat and J. Thielecke, "A Hidden Markov Model for pedestrian navigation.," in *Proceedings of the Workshop on Positioning, Navigation and Communication*, Dresden, March 2010.

[30] J. Liu, R. Chen, L. Pei, W. Chen, T. Tenhunen, H. Kuusniemi, T. Kröger, and Y. Chen, "Accelerometer assisted robust wireless signal positioning based on a hidden Markov model," in *Proceedings of the Position Location and Navigation Symposium (PLANS)*, Indian Wells, CA, USA, May 2010.

[31] H. Leppäkoski and J. Collin and J. Takala, "Pedestrian Navigation based on Inertial Sensors, Indoor Map, and WLAN Signals," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, March 2012, IEEE.

[32] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-Effort Crowdsourcing for Indoor Localization," in *Proceedings of the Mobicom*, Istanbul, August 2012, ACM.

[33] S. J. Ingram, D. Harmer, and M. Quinlan, "UltraWideBand Indoor Positioning Systems and their Use in Emergencies ," in *Proceedings of the Position Location and Navigation Symposium*, California, April 2004.

[34] L. Zwirello, T. Schipper, M. Harter, and T. Zwick, "UWB Localization System for Indoor Applications: Concept, Realization and Analysis," *Journal of Electrical and Computer Engineering*, vol. 2012.

[35] Y. Zhao, L. Dong, J. Wang, B. Hu, and Y. Fu, "Implementing Indoor Positioning System Via ZigBee Devices," in *Proceedings of the 42nd Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, October 2008.

[36] S. Fang, C. Wang, T. Huang, C. Yang, and Y. Chen, "An enhanced zigbee indoor positioning system with an ensemble approach," *IEEE Communications Letters*, vol. 16, no. 4, pp. 564–567, 2012.

[37] G. Fischer, B. Dietrich, and F. Winkler, "Bluetooth Indoor Localization System," in *Proceedings of the 1st Workshop on Positioning Navigation and Communication (WPNC 2004)*, Hannover, Germany, March 2004.

[38] C. Lee, Y. Chang, G. Park, J. Ryu, S. Jeong, S. Park, J. W. Park, H. C. Lee, K. Hong, and M. H. Lee, "Indoor positioning system based on incident angles of infrared emitters," in *Proceedings of the 30th Annual Conference of the IEEE Industrial Electronics Society*, Busan, Korea, November 2004.

[39] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik, "Indoor Localization without Infrastructure using the Acoustic Background Spectrum," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, Bethesda, Maryland, USA, June–July 2011.

[40] V. Filonenko, C. Cullen, and J. D. Carswell, "Indoor positioning for smartphones using asynchronous ultrasound trilateration," *ISPRS International J. Geo-Inf*, vol. 2, pp. 598–620, 2013.

[41] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 34, no. 3, pp. 334–352, August 2004.

[42] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, January 1992.

[43] R. Want and A. Hopper, "Active badges and personal interactive computing objects," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. 10–20, February 1992.

[44] "iBeacon technology," http://www.ibeacon.com/.

[45] N. E. Gemayel, S. Koslowski, and F. K. Jondral, "A low cost TDOA Localization System: Setup, Challenges and Results," in *Proceedings of the Workshop on Positioning, Navigation and Communication*, Dresden, March 2013.

[46] P. Stoica and J. Li, "Lecture notes: Source localization from range-difference measurements," *IEEE Singal Processing Magazine*, vol. 23, pp. 63–66, 2006.

[47] J. Yang and Y. Chen, "Indoor Localization Using Improved RSS-Based Lateration Methods," in *Proceedings of the GLOBECOM*. IEEE, November 2009.

[48] D. Dardari, U. Ferner, and M. Z. Win, "Ranging With Ultrawide Bandwidth Signals in Multipath Environments," in *Proceedings of the IEEE*, February 2009, vol. 97.

[49] A. Ward, A. Jones, and A. Hopper, "A new location technique for the active office," *IEEE Personal Communications*, vol. 4, no. 5, pp. 42–247, October 1997.

[50] R. M. Vaghefi, M. R. Gholami, and E. G. Ström, "Bearing-only target localization with uncertainties in observer position," in *Proceedings of the IEEE International PIMRC*, Istanbul, Turkey, September 2010.

[51] D. Hauschildt and N. Kirchhof, "Advances in thermal infrared localization: Challenges and solutions," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Zurich, September 2010.

[52] N. L. Dortz, F. Gain, and P. Zetterberg, "WiFi fingerprint indoor positioning system using probability distribution comparison," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2301–2304, March 2012.

[53] A. Kushki, K. N. Plataniotis, and A. N. Venetsapopoulos, "Kernel-based positioning in wireless local area networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 689–705, June 2007.

[54] J. Borenstein, "Internal Correction of Dead-reckoning Errors With the Smart Encoder Trailer," in *Proceedings of the International Conference on Intelligent Robots and Systems (lROS '94) - Advanced Robotic Systems and the Real World*, Munich, Germany, September 1994.

[55] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 328–342, June 1995.

[56] L. L. Menegaldo, G. A. N. Ferreira, M. F. Santos, and R. S. Guerato, "Development and navigation of a mobile robot for floating production storage and offloading ship hull inspection," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 9, pp. 3717–3722, September 2009.

[57] M. Mladenov and M. Mock, "A step counter service for Java-enabled devices using a built-in accelerometer," in *Proceedings of the of the 1st International Workshop on Context-Aware Middleware and Services*, Dublin, June 2009, ACM.

[58] K. Altun and B. Barshan, "Pedestrian dead reckoning employing simultaneous activity recognition cues," *Measurement Science and Technology*, vol. 23, no. 2, January 2012.

[59] W. Kang, S. Nam, and Y. Han, "Improved Heading Estimation for Smartphone-based Indoor Positioning Systems," in *Proceedings of the IEEE International PIMRC*, Sydney, Australia, September 2012.

[60] J. Jahn, U. Batzer, J. Seitz, L. Patino-Studencka, and J. G. Boronat, "Comparison and Evaluation of Acceleration Based Step Length Estimators for Handheld Devices," in *Proceedings of the Indoor Positioning and Indoor Navigation (IPIN)*, Zurich, September 2010, IEEE.

[61] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical Robust Localization over Large-Scale 802.11 Wireless Networks," in *Proceedings of the 10th annual International Conference on mobile computing and networking MobiCom*, Philadelphia, PA, USA, September 2004.

[62] J. Small, A. Smailagic, and D. P. Siewiorek, "Determining User Location For Context Aware Computing Through the Use of a Wireless LAN Infrastructure," 2000.

[63] K. Kaemarungsi and P. Krishnamurthy, "Properties of Indoor Received Signal Strength for WLAN Location Fingerprinting," in *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, Massachusetts, USA, August 2004.

[64] S. Beller, "Modelladaption zur Verbesserung von Fingerprinting basierter Indoornavigation," in *Master Thesis approved by the University of Paderborn*, Paderborn, July 2014.

[65] N. Singh, "Estimation of Parameters of a Multivariate Normal Population from Truncated and Censored Samples," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 22, no. 2, pp. 307–311, 1960.

[66] G. Lee and C. Scott, "Em algorithms for multivariate gaussian mixture models with truncated and censored data," *Computational Statistics & Data Analysis*, vol. 56, no. 9, pp. 2816–2829, September 2012.

[67] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[68] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density HMMs," *Computer Speech and Language*, vol. 9, pp. 171–185, 1995.

[69] M. J. F. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

[70] M. J. F. Gales and P. C. Woodland, "Mean and variance adaptataion within the MLLR framework," *Computer Speech and Language*, vol. 10, pp. 249–264, 1996.

[71] C. Drueke, "Navigation in Gebäuden mit Hilfe der Inertialsensorik eines Smartphones," in *Master Thesis approved by the University of Paderborn*, Paderborn, May 2013.

[72] "Website for android developers," http://developer.android.com/index.html.

[73] K. Hoang, S. Schmitz, C. Drueker, H. Tran, J. Schmalenstroeer, and R. Haeb-Umbach, "Server based Indoor Navigation using RSSI and Inertial Sensor Information," in *Proceedings of the Workshop on Positioning, Navigation and Communication*, Dresden, March 2013.

[74] K. Hoang and R. Haeb-Umbach, "Parameter Estimation and Classification of Censored Gaussian Data with Application to WiFi Indoor Positioning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, May 2013, IEEE.

[75] M. K. Hoang, J. Schmalenstroeer, C. Drueke, D. H. Tran Vu, and R. Haeb-Umbach, "A Hidden Markov Model for Indoor User Tracking based on WiFi Fingerprinting and Step Detection," in *Proceedings of the EUSIPCO*, Marrakech, September 2013, EURASIP.

[76] K. Hoang, J. Schmalenstroeer, and R. Haeb-Umbach, "Aligning Training Models with Smartphone Properties in WiFi Fingerprinting based Indoor Localization," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, April 2015, IEEE.

[77] "Wifarer company: Indoor positioning technologies," http://www.wifarer.com/.

[78] P. L. Bolliger, "Robust Indoor Positioning through Adaptive Collaborative Labeling of Location Fingerprints," in *A dissertation submitted to ETH Zurich*, Zurich, 2011.

[79] M. Boehner, "Optimizing the Energy-Consumption of WLAN-infrastructures," in *Master Thesis approved by the University of Paderborn*, Paderborn, October 2013.

[80] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, "Demystifying 802.11n Power Consumption," in *Proceedings of the 2010 International Conference on Power aware computing and systems*. ACM, 2010.

[81] "lighttpd project," http://www.lighttpd.net/.

[82] "FastCGI project," http://www.fastcgi.com/.

[83] "Postgis project," http://postgis.net/.

[84] "Postgres project," http://www.postgresql.org/.

[85] "Mapnik project," http://mapnik.org/.

[86] "JOSM: Java OpenStreetMap editor," https://josm.openstreetmap.de/.

[87] "Osm2pgsql: tool for converting OSM data to a PostgreSQL database," https://github.com/openstreetmap/osm2pgsql.

[88] S. Schmitz, "Serverstruktur einer WLAN-basierten Indoornavigation," in *Bachelor Thesis approved by the University of Paderborn*, Paderborn, October 2012.

List of own publications

[89] K. Hoang and R. Haeb-Umbach, "Parameter Estimation and Classification of Censored Gaussian Data with Application to WiFi Indoor Positioning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, May 2013, IEEE.

[90] M. K. Hoang, J. Schmalenstroeer, C. Drueke, D. H. Tran Vu, and R. Haeb-Umbach, "A Hidden Markov Model for Indoor User Tracking based on WiFi Fingerprinting and Step Detection," in *Proceedings of the EUSIPCO*, Marrakech, September 2013, EURASIP.

[91] K. Hoang, J. Schmalenstroeer, and R. Haeb-Umbach, "Aligning Training Models with Smartphone Properties in WiFi Fingerprinting based Indoor Localization," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, April 2015, IEEE.

[92] K. Hoang, S. Schmitz, C. Drueker, H. Tran, J. Schmalenstroeer, and R. Haeb-Umbach, "Server based Indoor Navigation using RSSI and Inertial Sensor Information," in *Proceedings of the Workshop on Positioning, Navigation and Communication*, Dresden, March 2013.