Telematics
Computer Systems

Freie Universität Berlin

Open Platform Architecture for Decentralized Localization Systems Based on Resource-Constrained Devices

DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

eingereicht beim Fachbereich Mathematik und Informatik
der freien Universität Berlin

vorgelegt von

Dipl. Ing. Zakaria Kasmi

Institute of Computer Science, Freie Universität Berlin, Germany

August 7, 2019

Erstgutachter: Prof. Dr.-Ing. Jochen Schiller
Zweitgutachter: Prof. Dr. Mesut Güneş

Tag der Disputation: 10.05.2019

À mes chers parents
(To my dear parents)

Dedicated to the loving memory of my grandmother Fatima Benjelloun & dear Rolf Reik.

Acknowledgments

After finishing this Ph.D. thesis, I am looking back to the previous years of research and I realize that it is impossible to thank everyone who influenced my research. I already apologize, since I will forget many persons. I extend my sincerest appreciation to all people and organizations who made this thesis possible.

First and foremost, my deepest gratitude goes to my advisor Prof. Dr.-Ing. Jochen H. Schiller, who offered me the invaluable opportunity of working as a research assistant at his chair to explore the fascinating world of sensor networks and Internet of Things as well as their localization algorithms. I want to thank him for his relief, valuable feedback, and words of encouragement. His support and guidance have been essential for the successful completion of this dissertation. His wisdom and generosity allowed me to research without restrictions and to have better access to well-founded knowledge. I am also thankful for his admirable example as a professor and successful computer scientist.

My deepest gratitude also goes to Prof. Dr. Mesut Güneş for being available as a reviewer and a supervisor for my thesis. My thanks to him for his advice and guidance during my first steps in research on wireless sensor networks.

I also want to thank my colleague Dr. Abdelmoumen Norrdine who helped me with my first steps in the world of localization systems and for providing a rich and fruitful environment for developing innovative ideas. I am very thankful for his input and interesting discussions. His expertise and experience, but also his kindness was invaluable to me. His symbiotic collaboration allowed creating and maintaining the RcdMathLib, which is becoming a great contribution to the research world of localization algorithms for resource-constrained devices. Big thanks go also to Dr.-Ing. Hendrik Hellmers for the cooperation to deploy the platform architecture for accurate 3D positioning for a mobile platform in non-line-of-sight scenarios.

I want to thank my colleagues from the projects AVS-Extrem and VIVE: Dr. Norman Dziengel, Marco Ziegert, Stephan Adler, Stefan Pfeiffer, and Martin Seiffert for their passion, helpfulness, and support on many low-level aspects. I would also like to thank my colleagues from the project SAFEST: Prof. Dr. Matthias Wählisch and Alexandra Danilkina for their valuable advice and the wonderful team play. It was an immense pleasure to work with you.

I would like to thank all the many friendly colleagues I was able to work with at the Institute of Computer Science, Freie Universität Berlin. I also thank Naouar Guerchali for the support in the implementation and evaluation of the Levenberg–Marquardt algorithm.

I am thankful for all the fantastic work that has been done by the amazing community on

RIOT.

Many thanks are also due to my proofreaders Mandy Perry and William Mack Perry, who patiently reviewed the thesis, but especially to Dr. Christel J. Guillory, who thoroughly reviewed the whole thesis.

This work is dedicated to my mother and my father. They always encouraged me to give my best and infused me with boundless curiosity and persistence. Thank you for your absolute confidence in me and for your understanding and support for my pursuits.

Berlin, July 2019
Zakaria Kasmi

Abstract

A platform architecture for positioning systems is essential for the realization of a flexible localization system, which interacts with other systems and supports various positioning technologies and algorithms. The decentralized processing of a position enables to push the application-level knowledge into a mobile station and avoids the communication with a central unit such as a server or a base station. In addition, the calculation of the position on low-cost and resource-constrained devices presents a challenge due to the limited computing and storage capacity as well as power supply. Therefore, this thesis proposes a platform architecture which allows for the design of a system with the following advantages: reusability of the components, extensibility (e.g., with other positioning technologies) and interoperability. Furthermore, the position is computed on-the-fly on a low-cost device such as a microcontroller, which simultaneously performs additional tasks such as data collecting or preprocessing based on an operating system. The platform architecture is designed, implemented, and evaluated based on two systems: a time-of-arrival and a field strength-based positioning system. These systems use Ultra-Wideband (UWB) and Direct Current (DC)-pulsed magnetic signals, respectively. Suitable algorithms are proposed to compute an unoptimized position (start position). These algorithms will be analyzed and compared with respect to the stability, efficiency, complexity, and memory requirements. An adaptive algorithm is developed for the optimization of the position, which is based on the Singular Value Decomposition (SVD), Levenberg–Marquardt (LVM) algorithm, and the Position Dilution of Precision (PDOP). This algorithm allows an adaptive selection mechanism for the LVM algorithm. This adaptive algorithm enables saving of resources such as memory, computing time, and energy on resource-constrained devices. Furthermore, two variants of the LVM algorithms are used: the Dahmen-Reusken LVM and Madsen LVM, which are analyzed and compared with the Gauss-Newton algorithm. All the algorithms are derived in a convenient form for resource-constrained devices. Since the parameters of the LVM algorithm impact the accuracy as well as the required iteration number, the influence and the choice of the right parameter combination will be determined, analyzed and discussed. Finally, a method is designed and evaluated to reduce multipath errors on the mobile station. This method allows an accurate localization in non-line-of-sight scenarios.

Contents

List of Figures

List of Tables

Acronyms

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks . . . 2
AC Alternating Current . . . 20
ADC Analog to Digital Converter . . . 25
AL Application Layer . . . 4
ANSI American National Standards Institute . . . 30
AOA Angle of Arrival . . . 9
AP Access Point . . . 36
API Application Programming Interface . . . 30
CAN Controller Area Network . . . 26
CDF Cumulative Distribution Function . . . 43
CDMA Code Division Multiple Access . . . 62
CDU Coil Driver Unit . . . 60
CFS Completely Fair Scheduler . . . 70
CGS Classical Gram–Schmidt . . . 104
CPU Central Processing Unit . . . 16
CSMA/CA Carrier Sense Multiple Access/Collision Avoidance . . . 39
CSS Chirp Spread Spectrum . . . 39
DC Direct Current . . . 20
DOA Direction of Arrival . . . 2
DOP Dilution of Precision . . . 91
DQD Differential Quotient Difference . . . 104
DR Dead Reckoning . . . 13
DR-LVM Dahmen–Reusken–Levenberg–Marquardt . . . 3
DSO Digital Sampling Oscilloscope . . . 79
DSSS Direct Sequence Spread Spectrum . . . 39
ECDF Empirical Cumulative Distribution Function . . . 114
FCC Federal Communications Commission . . . 15
FDMA Frequency Division Multiple Access . . . 61
FIFO First-In, First-Out . . . 28
FROB Frobenius metric . . . 43
GDOP Geometric Dilution of Precision . . . 91
GE Gaussian Elimination . . . 104
GKR Golub–Kahan–Reinsch . . . 104
GN Gauss–Newton . . . 4
GNM Gauss–Newton Method . . . 3
GNSS Global Navigation Satellite System . . . 1
GPIO General-Purpose Input/Output . . . 25
GPS Global Positioning System . . . 1
GSM Global System for Mobile Communications . . . 7
HDOP Horizontal Dilution of Precision . . . 91

HF High Frequency . . . 58
HMM Hidden Markov Model . . . 37
I2C Inter Integrated Circuit . . . 42
ID Identification . . . 7
IEEE Institute of Electrical and Electronics Engineers (I Triple-E) . . . 39
ILS Indoor Localization System . . . 35
IMU Inertial Measurement Unit . . . 38
INS Inertial Navigation System . . . 38
IoT Internet of Things . . . 2
IP Internet Protocol . . . 24
IPC Inter-Process Communication . . . 27
IPv4 Internet Protocol version 4 . . . 45
IPv6 Internet Protocol version 6 . . . 151
IR Impulse Radio . . . 39
ISR Interrupt Service Routine . . . 30
IT Information Technology . . . 23
JSON JavaScript Object Notation . . . 42
LBS Location-Based Service . . . 1
LON Local Operating Network . . . 26
LPS Local Positioning System . . . 1
LSD Large-Scale Direct . . . 38
LTE Long-Term Evolution . . . 8
LTS Least Trimmed Squares . . . 113
LVM Levenberg–Marquardt . . . 3
MAC Medium Access Control . . . 30
MAD Median Absolute Deviation . . . 42
MCU Microcontroller Unit . . . 4
MDDM Multipath Distance Detection and Mitigation . . . 113
MGS Modified Gram–Schmidt . . . 104
MILPS Magnetic Indoor Local Positioning System . . . 4
ML Multilateration . . . 153
M-LVM Madsen–Levenberg–Marquardt . . . 3
MS Mobile Station . . . 2
NLoS Non-Line-of-Sight . . . 8
NLS Non-linear Least Squares . . . 51
OS Operating System . . . 23
PC Personal Computer . . . 19
PCN People-Centric Navigation . . . 38
PDF Probability Density Function . . . 43
PDOP Position Dilution of Precision . . . 3
PDR Pedestrian Dead Reckoning . . . 38
PMA Propagation Model Algorithm . . . 8
POSIX Portable Operating System Interface . . . 29
Profibus Process Field Bus . . . 26
QF Quality Factor . . . 69
RAM Random-Access Memory . . . 25
RcdMathLib Open-Source Mathematical Library for Resource-Constrained Devices . . . 46

RF	Radio Frequency	26
RFID	Radio Frequency Identification	15
RMS	Root Mean Square	21
RMSE	Root Mean Square Error	43
ROM	Read-Only Memory	67
RoPEUS	Robust Position Estimation in Ultrasound Systems	113
RS	Reference Station	12
RSS	Received Signal Strength	8
RSSI	Received Signal Strength Indicator	1
RTC	Real-Time Clock	59
RTLS	Real-Time Locating System	17
RTT	Round-Trip Time	12
SHM	Structural Health Monitoring	23
SL	System Layer	4
SLAM	Simultaneous Localization and Mapping	38
SPI	Serial Peripheral Interface	25
SRAM	Static Random-Access Memory	72
SSR	Solid State Relay	72
SVD	Singular Value Decomposition	3
TCXO	Temperature Compensated Crystal Oscillator	71
TDMA	Time Division Multiple Access	4
TDOA	Time Difference of Arrival	10
TDOP	Time Dilution of Precision	91
TOA	Time of Arrival	1
TR	Time Reversal	17
UART	Universal Asynchronous Receiver Transmitter	25
UC	University of California	28
UDP	User Datagram Protocol	151
UMTS	Universal Mobile Telecommunications System	7
UWB	Ultra-Wideband	2
VDOP	Vertical Dilution of Precision	91
WLAN	Wireless Local Area Network	2
WSN	Wireless Sensor Network	15
ZUPT	Zero Velocity Update	38

CHAPTER 1

Introduction

Nowadays, the accurate localization of a user or an object is indispensable for Location-Based Services (LBSs), such as asset tracking, inventory management, or routing and navigation. In recent years, location sensing systems have become popular and emerged as a vital research area. Therefore, research and commercial products have been developed in academic as in industrial contexts. Localization systems use various sensors and communication technologies as well as approaches to locate objects or persons [1].

Global Positioning System (GPS) and similar systems play important roles in applications such as car navigation or vehicle tracking and monitoring. GPS accuracy is restricted in urban areas due to multipath propagation errors as well as limited visibility of the satellites, especially with narrow streets and tall buildings [2]. Furthermore, although the use of pseudo-lites, which generate and transmit GPS-like signals, might enable GPS to work indoors; pseudo-lites based solutions must cope with several issues such as multipath effects, precise synchronization, and government restrictions [3, 4].

Since GPS does not work in indoor environments, alternative localization techniques have been developed for indoor positioning. Deak et al. present a survey on indoor localization systems, which use different technologies such as Time of Arrival (TOA) or Received Signal Strength Indicator (RSSI) [5]. Most indoor localization systems necessitate that tags/electronic devices must be mounted on objects or carried by the person being tracked to estimate their localization [5]. Common features of indoor localization systems are [6]:

- *Deployment environment:* The area where the system is used should be prepared by installing fixed reference units, which interact with a mobile target to estimate its location. Environmental conditions, such as walls, result in various grades of attenuation, scattering, and reflection. These conditions should be considered by the selection or design of the localization technology or system, respectively.
- *System range:* Indoor localization systems are usually limited to a small area and, therefore, are referred to as *Local Positioning Systems (LPSs)*. In contrast, the 2G/3G cellular network positioning systems, or the Global Navigation Satellite Systems (GNSSs) belong to global coverage localization systems.
- *Geometric location model:* The location information can be represented by using a two- or three-dimensional geometric model such as a local Cartesian or a polar coordinate system. A logical location model can be more appropriate in context-based applications:

Given a room, which can be labeled as a kitchen or living room, where a person or an object is located. Various positioning, as well as mobile computing applications, take advantage of knowing a user's logical location such as a coffeehouse or a restaurant. This can allow a shopkeeper to send an electronic coupon, for purchasing coffee or food, to a customer entering the store [7].

- *Localization techniques:* They can be divided into various classes [8]. Based on the type of the distance measurement method, they can be classified as range-free or range-based location systems. As a function of the implementation of the system, they can also be classified as hardware-based and software-based positioning systems. Depending on the system architecture, they can be grouped into tightly or loosely coupled localization systems [9].

1.1. Research Objectives and Contributions

A platform architecture is required for the realization of the previously-mentioned localization systems. To the best of our knowledge, there is neither a standard for the design, nor a detailed description of the architecture of a positioning system. The architecture of most localization systems is roughly described and divided into two parts: the sensor hardware and the positioning algorithm [10]. The first part relies on a variety of technologies, such as electromagnetic waves (e.g., Ultra-Wideband (UWB) or Wireless Local Area Network (WLAN)), or ultrasound. The positioning algorithm is based on various signal measurement methods, such as TOA, RSSI or Direction of Arrival (DOA) [10, 11].

The goal is to design and validate an open platform architecture that can be implemented on resource-constrained devices such as microcontrollers. These devices can build an Internet of Things (IoT) of networked embedded objects. The combination of real-time localization, Internet, and embedded sensors allows transforming everyday objects into smart objects that can interpret, perceive, and interact with the environment. The platform architecture should be open in the sense of being able to communicate and interact with other systems as well as being open source. The platform allows for decentralized, continuous (smooth), and accurate localization.

In this thesis, the focus is on anchor-based indoor positioning systems. The main contributions of this work are:

1. Proposal of a layer- and modular-based architecture, which enables on-the-fly location calculation on the Mobile Station (MS). The position is computed on low-cost, resource-constrained devices such as microcontrollers. The proposed platform is non-proprietary and open, since it can be easily extended with other sensors to enable the implementation of positioning systems based on other technologies. The proposed platform can interoperate with existing systems and protocols such as IPv6 over

Low-Power Wireless Personal Area Networks (6LoWPAN).

2. Development of a preprocessing method to remove outliers in measured data, which is convenient for resource-constrained devices with limited stack memory.
3. Introduction of a method that calculates the 3D position based on the Singular Value Decomposition (SVD).
4. Proposition of a pre-processed method for the localization calculation, avoiding the execution of memory and computationally-expensive algorithms such as the SVD or Moore–Penrose, on resource-constrained devices.
5. Demonstration of the feasibility to deploy the Moore–Penrose algorithm, which is based on SVD, on resource-constrained devices.
6. Derivation of the Gauss–Newton as well as the Levenberg–Marquardt algorithms in a convenient form for resource-constrained devices to improve the position estimate.
7. Development and exploration of an adaptive algorithm for saving resources such as memory, computing time, and energy on resource-constrained devices. This adaptive algorithm is based on the SVD, Levenberg–Marquardt (LVM) algorithm and the Position Dilution of Precision (PDOP). Furthermore, two variants of the LVM algorithms are used: the Dahmen–Reusken–Levenberg–Marquardt (DR-LVM) and Madsen–Levenberg–Marquardt (M-LVM), which are analyzed and compared with the Gauss–Newton Method (GNM). All algorithms are derived in a convenient form for resource-constrained devices. Since the parameters of the LVM algorithm impact the accuracy as well as the required iteration number, the influence and the choice of the right parameter combination will be determined, analyzed and discussed.
8. Design of a multipath detection and mitigation algorithm in a suitable way for resource-constrained devices. This algorithm is implemented as well as evaluated in a simulated and real environment to demonstrate that the mobile station can operate efficiently in non-line-of-sight environments.
9. Achievement of the open source of the suggested platform architecture by integrating into RIOT-OS [12], a software library for performing numerical linear algebra, positioning algorithms, and signal processing on resource-constrained devices.

These contributions are published in [13, 14, 15]. Other contributions related to the dissertation are published in [16, 17, 18, 19, 20].

1.2. Thesis Outline

The remainder of the thesis is organized as follows:

Localization: Chapter 2 gives a brief overview of the measurement methods used as well as the current state of the art of localization systems. Furthermore, both the UWB and the magnetic-based positioning systems are compared.

Operating Systems for Resource-Constrained Devices: In Chapter 3, first the anatomy of resource-constrained devices is presented. Then, various operating systems for resource-limited devices are compared. Finally, the RIOT-OS, which is a small, open-source operating system for memory-constrained systems with a focus on low-power wireless IoT devices, is introduced.

Open Platform for Positioning Systems: In Chapter 4, the author compares various indoor localization systems from the industry as well as from the research area in terms of the presented architecture and platform. He also provides a brief overview of existing standards for localization systems. A layered, open platform for decentralized positioning systems is introduced by describing the layers, the components, and their functions. The platform is comprised of two main layers: The System Layer (SL) and the Application Layer (AL). Both layers are discussed within this chapter.

Decentralized UWB-Based Indoor Localization System: In Chapter 5, a decentralized localization system using UWB signals is introduced. Based on Chapter 4, the proposed architecture is explored by implementing both the SL and AL. Suitable algorithms are introduced in the AL to meet requirements such as limited stack size, computing capacity in addition to the trade-off of the convergence time versus the accuracy by resource-constrained devices. Furthermore, the UWB-based localization system is evaluated on a Microcontroller Unit (MCU) in a real-world scenario and in terms of complexity, position accuracy, computing time, as well as energy consumption.

Decentralized Magnetic Indoor Positioning System: The proposed architecture is investigated by discussing and implementing all layers for the Magnetic Indoor Local Positioning System (MILPS) in Chapter 6. The entire system is synchronized based on the Time Division Multiple Access (TDMA) approach, which enables a stand-alone control of coils. Additionally, the TDMA-based method allows the MS a correct assignment of the magnetic field data generated by the coils. The localization and the optimization algorithms are proposed in a convenient form for resource-scarce devices. Finally, the system is evaluated based on a resource-constrained device, which is an MCU, in a real-world scenario and in terms of complexity, position accuracy, computing time, and energy consumption.

Algorithms and Position Optimization: Suitable algorithms are investigated and analyzed to compute a position on resource-constrained devices. These algorithms are analyzed with respect to stability, complexity, and memory requirements. The Gauss–Newton (GN) as well as the LVM algorithms, which are derived in a convenient form for resource-constrained devices, are used for the position optimization. Two variants of the LVM methods are analyzed and compared with the GNM. An adaptive optimization algorithm is developed based on the SVD algorithm and the PDOP to allow an adaptive

selection mechanism for the LVM algorithm. The adaptive selection mechanism enables saving memory, computing time, and energy on resource-constrained devices. Furthermore, the parameters of the LVM algorithms are analyzed, because they impact the number of iterations and the accuracy of the position. The choice of the correct parameter combination is also analyzed and discussed. Finally, an algorithm is designed and implemented in an appropriate way for resource-limited devices to reduce the multipath effects on the mobile station, and so enables an efficient localization in non-line-of-sight scenarios.

Example Applications: The proposed architecture is deployed in three real-world scenarios. In the first scenario, an application that enables the localization of objects or persons inside a building is developed. In the second scenario, moving persons are tracked by using MILPS as well as an additional method to overcome the impact of spatially varying ambient magnetic fields. In the third scenario, a robot is tracked by combining MILPS with inertial measurement units. All these applications are described in Chapter 8.

Conclusion and Outlook: The thesis is concluded by outlining the results of theoretical investigations and real-world deployments to resume with the systems' adaptability and accuracy in Chapter 9. The thesis is finalized with an outlook of possible applications and extensions of the proposed platform architecture.

CHAPTER 2

Localization

Humans always have the need to know location information. In early childhood, children explore their environment by developing a sense of time, shape, space, and place [21]. In adulthood, they noticed the way to go hunting or how to reach basic resources such as food or water. In order to answer the fundamental questions "Where am I ?" or "Where is it ?", people developed various strategies by memorizing specific objects or/and features of the environment: For example, mountains, unusually shaped plants or trees, or buildings such as a church. For hundreds of years, sailors used the position of celestial bodies such as planets or stars, the moon, or the sun, to navigate the sea. Nowadays, many technologies exist that allow an automatic determination of the position of persons or objects. Well-established technologies such as radar or GNSS rely on one or more sensor measurements, a common process used by all localization technologies. A critical issue in localization technology are measurement techniques, which will be discussed in the next Section 2.1.

2.1. Measurement Methods

Localization methods are based on various signal measurement approaches such as TOA, DOA or RSSI. Generally, these methods involve the transmission and reception of signals between one or more hardware devices [10]. In the following subsections, commonly used measurement methods for localization will be presented.

2.1.1. Proximity Location Sensing

The proximity location method provides the location of an object in relation to an area or a known position [22]. The object is captured by using a physical measurement technique, which enables sensing the object's presence within a limited range. Although the proximity localization delivers coarse-grain information, it can be sufficient in some cases, for example, to provide a statement about the room where a person is locating.

Methods for sensing proximity can be generally categorized into three classes [23]: wireless or mobile cellular-based, Identification (ID), or physical sensing methods. In the first class, the mobile object can be localized, if it is within the reach of one or more access points of a wireless or mobile cellular network [24]. Mobile cellular network systems can be based on a Global System for Mobile Communications (GSM), Universal Mobile Telecommunications

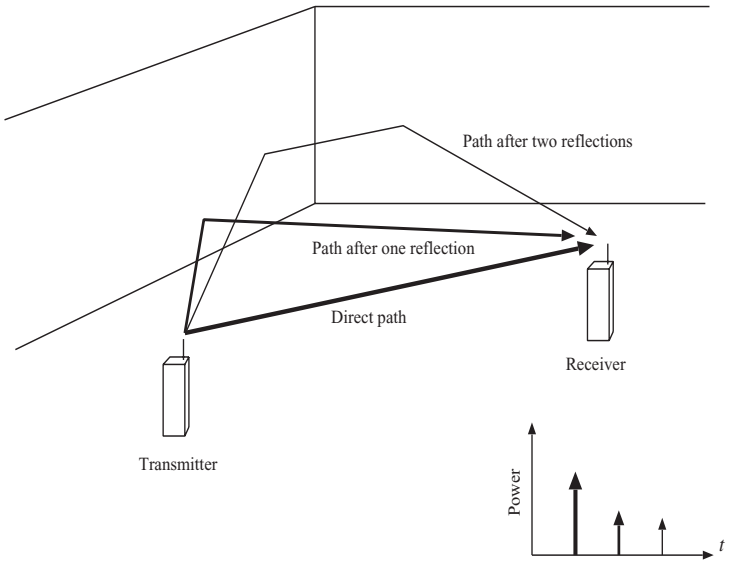


Figure 2.1.: *A typical indoor scenario in which the transmitted pulse is reflected, figure adapted from [29]*

System (UMTS), or Long-Term Evolution (LTE) network [25, 26, 27]. The second class of proximity sensing relies on ID systems, which, for example, interrogate a tag or scan a label. The tags or labels can be either attached to a person or mounted on objects. Therefore, their positions can be approximately localized, because ID systems such as barcode identification systems have a known position. By using a physical sensing method, the contact with an object is detected by means of touch or pressure sensors, or capacitive field detectors [23]. Partridge *et al.* presented a contact system that enables communication between the objects being touched by a user as well as their identifications [28].

2.1.2. Received Signal Strength-Based Method

The position of a target can be estimated by measuring the strength of a signal originating from various transmitters. The Received Signal Strength (RSS) method can use distances between the target and the transmitter given that the attenuation of the received signal is inversely proportional to the distance. In this case, the measured distances are affected by factors such as Non-Line-of-Sight (NLoS), signal scattering, or multipath reflections, which can intensify in an indoor environment due to the existence of obstacles such as furniture or walls (see Figure 2.1). In the case of the line of sight, the electromagnetic signal spreads out directly in a straight path from an emitter to a receiver and shows higher energy compared with reflected signals (*cf.* Figure 2.1). Mapping RSS values to distances is called a Propagation Model Algorithm (PMA). Alternatively, the RSS method can also use a fingerprinting algorithm, which will be discussed in the next Section 2.1.3.

2.1.3. Fingerprinting

The fingerprinting technique matches the features extracted from the sensor data against location information, whereby the features can be the RSS or the light characteristics. The fingerprinting approach can be divided into two classes: static or differential fingerprinting [30]. With the static fingerprinting, the observed features are associated with an object location [30]. However, with differential fingerprinting, the location is estimated by tracking the difference between successive observation sets [30]. Location fingerprinting involves two phases: the off-line and the on-line phase. In the first phase, also called the calibration phase, a calibration function is created. The calibration function can be a radio map holding the RSS values of the radio signals as a function of the localization [31]. In the online phase, called the location estimation phase, the position is estimated based on the measured signal strengths and the calibration function.

2.1.4. Angle-Based Method

The localization can be performed using angles that are relative to multiple reference points [10]. This technique is called **angulation** since angles are used between a target and the anchors.

The two-dimensional position of an object can be estimated by applying trigonometry to the measured angles (α and β) and the distance (d) between two reference points A and B (see Figure 2.2). In a the three-dimensional case, an azimuth measurement is additionally needed for one distance and two angle measurements. This approach is called the Angle of Arrival (AOA) in the literature.

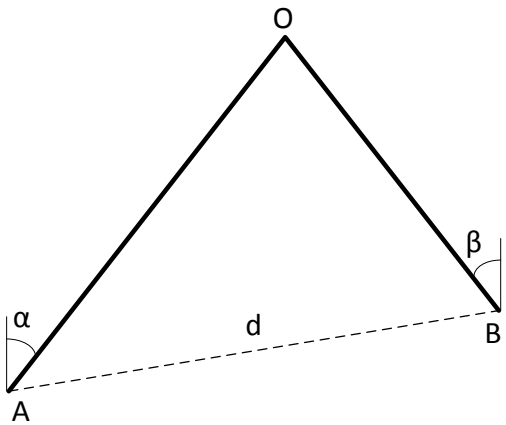


Figure 2.2.: *Measurement method based on angulation*

The angle can be determined by using an antenna array that is composed of adjacent reference points equipped with antenna elements [32]. The angle α of the signal arriving at the antenna array is measured based on the time difference of the wave signal at the

stations S_1 and S_2 and the reference direction, which is normal to the line a (see Figure 2.3).

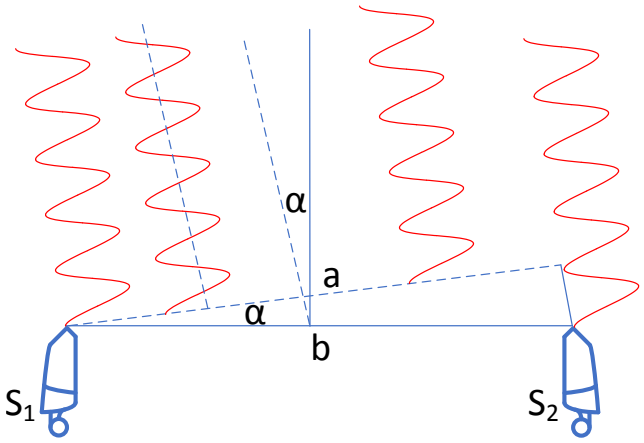


Figure 2.3.: *Measurement of the angle based on antenna array. S_i, Antenna i*

The angle α can be calculated as follows:

$$\alpha = \arcsin\left(c \cdot \frac{\Delta t}{b}\right), \tag{2.1}$$

where c is the velocity of light, Δt is the time difference, and b is the distance between two antenna elements.

2.1.5. Distance-Based Method

Distance-based methods rely on distance measurements between a transmitter and a receiver, whereby the distance information can be obtained by the RSSI, AOA, TOA, or Time Difference of Arrival (TDOA). This approach is called **lateration** due to the use of distances between the localization entities.

2.1.5.1. Received Signal Strength Indicator

RSSI methods are based on the signal attenuation model, whereas the distance is resolved by measuring the received signal strength (P_{rx}) if the transmission power (P_{tx}) is known. The distance d between the transmitter (T_x) and the receiver (R_x) can be calculated by using the non-logarithmic loss-path equation as follows [33]:

$$P_{rx} = k \cdot \frac{P_{tx}}{d^{\alpha}} \tag{2.2}$$

$$d = \sqrt[\alpha]{\frac{k \cdot P_{tx}}{P_{rx}}}, \tag{2.3}$$

where k is a constant and α is the path loss exponent.

A corresponding and common model to estimate a distance between a T_x and R_x is the log-distance path loss model, which presupposes that the T_x and R_x are equipped with identical antennas and the reference distance d_0 is known. Since, the mean path loss increases exponentially with the distance [34, 35]:

$$\overline{PL} \propto \left(\frac{d}{d_0}\right)^{\alpha}, \tag{2.4}$$

the mean path loss, in decibels, is defined as follows:

$$\overline{PL}\,(d)\,[dB] = PL\,(d_0)\,[dB] + 10 \cdot \alpha \cdot \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma, \tag{2.5}$$

where the $PL\,(d_0)$ is the path loss from the transmitter to the reference distance, the logarithmic term is the path loss described by (2.4), and X_σ represents the log-normally variation in the channel path loss, which is a zero mean Gaussian random variable. The path loss exponent (α) indicates the signal drop over the distance in various environments (see Table 2.1).

Table 2.1.: *Typical values of path loss exponent; source: [36]*

Environment	Path Loss exponent
Free Space	2
Urban Area	2.7 to 3.5
Shadowed Urban Area	3 to 5
Indoor (line-of-sight)	1.6 to 1.8

2.1.5.2. Time of Arrival

The TOA is the most popular used measurement technique, which is based on the time measurement of a radio signal between a transmitter and receiver. The ranging method can be classified into the synchronous and asynchronous approach, whereas a global synchronization is required by the synchronous method [37]. In this case, the synchronization is usually accomplished by using expensive but precise oscillators and the ranging process is one-way [38]. By contrast, a global synchronization is not necessary using the asynchronous method, since each ranging instance uses its own clock to measure the propagation time between two ranging instances [39, 40]. In this case, the ranging is two-way or bidirectional [37]. The TOA can be estimated by various ranging techniques, which one- and two-way ranging are the main techniques [41]. Figure 2.4(a) illustrates the one-way TOA, whereby a time-stamp is included in the generated signal from the transmitter T_x. The time of flight and the distance can be calculated at the receiver R_x, respectively, as follows:

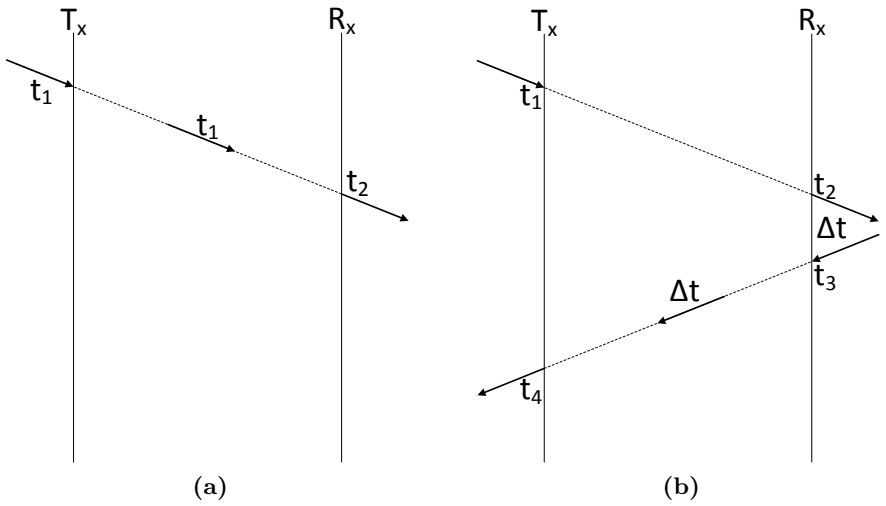


Figure 2.4.: *Time of Arrival: (**a**) One-way Ranging, (**b**) Two-way Ranging*

$$t_{owr} = t_2 - t_1, \tag{2.6}$$

$$d = c \cdot t_{owr}, \tag{2.7}$$

where the c is the speed of light.

In contrast, in case of the two-way ranging, the signal Round-Trip Time (RTT) is estimated by sending a signal from the transmitter T_x that is replied by the receiver (see Figure 2.4(b)). Analogously, the time of flight and the distance can be calculated at the transmitter T_x, respectively, as follows:

$$t_{twr} = t_1 - t_4 - \Delta t, \tag{2.8}$$

$$\Delta t = t_3 - t_2, \tag{2.9}$$

$$d = \frac{c}{2} \cdot t_{twr}, \tag{2.10}$$

2.1.6. Hybrid Methods

Hybrid measurement techniques use a combination of range or angle methods. Examples of these methods are the TOA/AOA, TDOA/AOA or the TOA/RSS. Hybrid methods enable an accurate determination of a position and a more reliable localization estimation. For example, they can be beneficial if the number of Reference Stations (RSs) are limited [42]. The TOA can be combined with the AOA to improve the accuracy of a target position as well as to eliminate the NLoS effects by using possible intersections of the TOA circles and AOA line to the MS [43]. The location estimation can also be enhanced by mixing and

combining the TOA and AOA. In this case, the best position outcome is selected from the AOA, TOA, and the TOA/AOA methods based on the mean and variance of the distance between each estimated position and their mean [44].

2.1.7. Dead Reckoning

Dead Reckoning (DR) is the oldest method of navigation that was used by ancient sailors to find their way at sea. The estimation of a new (future) position is calculated by combining compass heading, knowledge of sea currents, and the vessel speed. The speed of the vessel is determined by measuring the time taken by an object thrown overboard to travel a fixed distance along the side of the ship [45]. In modern systems, inertial sensors such as digital accelerometers, magnetometers, and gyroscopes are used to realize dead reckoning approaches. DR can be used for robot navigation, pedestrian-tracking, and autonomous navigation. DR is the process of estimating a current position by projecting heading and speed from a past position. The heading and speed are combined to a movement vector ($\vec{v_1}$) that represents the position change from a known position $\vec{x_0}$ to an estimated position $\vec{x_1}$ [45]. The heading is commonly measured by using magnetic sensors and a gyroscope. The speed is usually provided by speed sensors already integrated into vehicles such as automobiles, boats, or aircrafts. In contrast, users of wearable devices such as smartphones are not equipped with such sensors. In this case, the speed can be estimated by using pedometers or accelerometers [45].

2.2. Localization Systems

The exponential growth in information and communication technology, the increasing role of ubiquitous computing, as well as semantic-oriented information and location data mining tasks, have resulted in substantial business interests in LBSs [46, 47, 48]. The localization systems can be deployed in outdoor as well as in indoor environments. The outdoor localization systems are deployed outside buildings and usually use the GPS technology. The indoor localization systems are used inside buildings such as hospitals or airports; whereby indoor localization applications and technologies enable an automatic positioning of persons or objects inside buildings and provide context-dependent information on a mobile device. Examples of indoor LBSs are position assignment of products inside a warehouse and the navigation to the right platform or gate at train stations or airports [49]. Figure 2.5 illustrates a system for the localization of fire fighters. This positioning system uses the GPS and UWB for outdoor and indoor localization, respectively.

The surrounding environment characterizes a positioning system and determines its constraints and expected performance. Theoretically, positioning systems can be deployed both outdoors and indoors, but their efficiency differs greatly from each other, because indoor

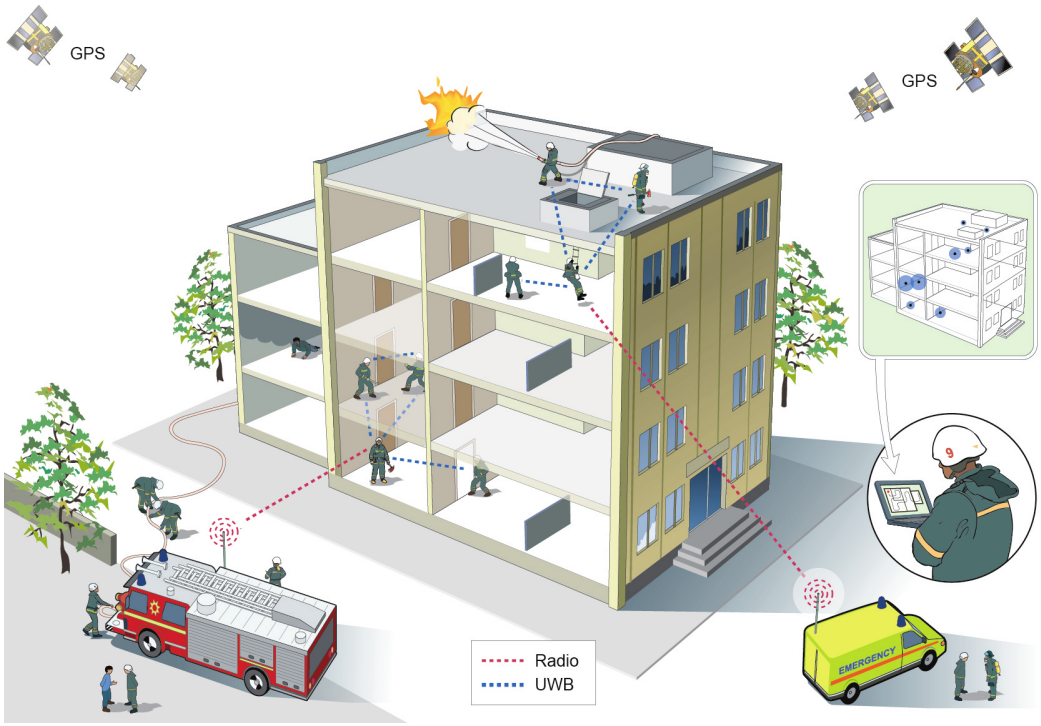


Figure 2.5.: *Example of real-world localization system, figure adapted from [46]*

surrounding areas raise a challenge for determining a position, especially for systems based on wireless technologies. This occurs due to factors such as signal scattering and attenuation because of the high density of obstacles, multipath reflections from walls and furniture, NLoS and environmental changes due to opening doors and moving people [49]. Another challenge is the position finding in a harsh environment and severe industrial conditions, like a bunker, coal mine or locating a firefighter in a hazardous area (*see* Figure 2.5).

Positioning systems can be classified into two groups depending on the infrastructure: infrastructure-less and infrastructure-based positioning systems. Examples of infrastructure-less methods are video-based (e.g., smart-phone camera [50], robot navigation [51, 52]), speech source based, or wireless sensor network-based methods. In contrast, infrastructure-based positioning systems need an infrastructure such as permanently installed hardware, electrical power, walls or tripods for reference points mounting, or Internet access [49]. Infrastructure-based positioning systems are, for example, UWB- or magnetic field-based localization systems [53, 13]. The last two classes of positioning systems can complement each other. The infrastructure-based positioning system can be used as a complementary system in an infrastructure-less positioning system to provide a starting point or to support a long-term stability. On the other hand, the infrastructure-less based method can cover areas, which cannot be reached by an infrastructure-based method [54].

Another class of positioning systems are the collaborative and device-free localization

systems. The collaborative position localization technique is based on various units (sensor nodes) that collaborate to determine their positions. This technique is also called a cooperative or network position location [11]. The collaboration between sensor nodes has normally been assumed to occur in Wireless Sensor Networks (WSNs) [55]. Device-free positioning systems, which can be used for perimeter security, enable the tracking of users without wearing any devices. Device-free localization systems can use technologies such as pressure or electric field sensors installed under the floor [56].

In the past few years, numerous technologies have been evaluated for positioning and navigation tasks inside buildings. These technologies are based on several physical principles and exhibit different performance characteristics. The physical layer can use a variety of technologies, such as UWB [57, 58] or WLAN [59], which are based on electromagnetic waves. Other localization systems are based on ultrasound [60], infrared [24], Radio Frequency Identification (RFID) [61], Bluetooth [62] or computer vision techniques [63]. The main drawbacks of these systems are signal propagation errors due to attenuation, shadowing, multipath and signal delay inside buildings or poor lighting conditions. Even if some technologies like UWB are more robust against the mentioned effects, it is impossible to suppress signal propagation errors completely. However, in contrast to electromagnetic waves, magnetic signals can pass through any building material without significant attenuation or distortion and, are generally suitable for indoor positioning purposes. We restrict our focus to UWB and the magnetic field technologies, since they are well suited for indoor localization applications. We will provide a brief review about them in the next subsections.

2.2.1. Ultra-Wideband Signals

UWB signals occupy a wide band of frequency and are defined as signals with "*a fractional bandwidth of larger than* 20 *% or an absolute bandwidth of at least 500 MHz*" [64]. Therefore, UWB signals are characterized by very short pulses and consequently a large bandwidth that enables the reduction of the multipath effects as well as an accurate measurement of the TOA between two UWB transceivers. These features facilitate the implementation of accurate indoor localization systems. Furthermore, they enable high data rates (up to 100 Mbps) for near-field data transmission [29]. Figure 2.6 illustrates a UWB pulse in the time and frequency domain, respectively. The UWB pulse shape is known as a Gaussian doublet, which is typical of a nanosecond or picosecond order (see Figure 2.6(a)) [29]. The spectrum of the UWB pulse signal is illustrated in Figure 2.6(b), whereby the center frequency is about 5 GHz with a 3 dB bandwidth

The Federal Communications Commission (FCC) sets a power requirement of 41.3 dBm/MHz, equal to 75 nW/MHz for UWB signals, which allows them to reside below the noise floor and coexist with current radio services with minimal or no interference (see Figure 2.7) [65]. In addition, this power restriction puts UWB systems in the class of unin-

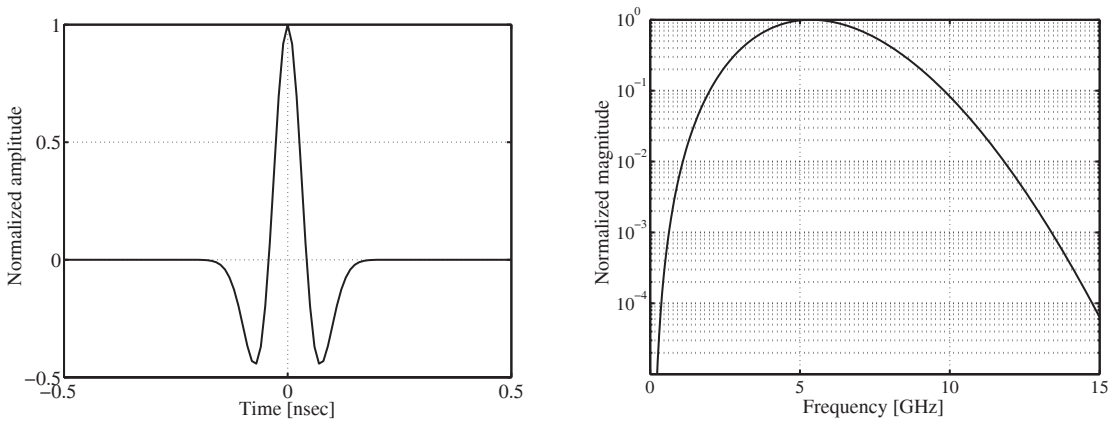


Figure 2.6.: *Idealized (**a**) UWB pulse and (**b**) the spectrum of an UWB pulse, adapted from [29]*

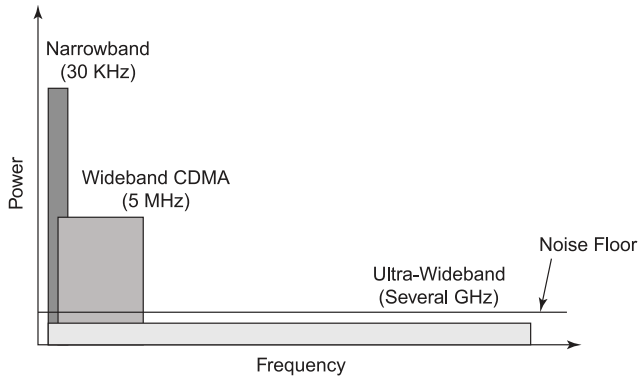


Figure 2.7.: *Coexistence of UWB signals with narrowband and wideband signals [65]*

tentional radiators such as Central Processing Unit (CPU) boards or computer monitors.

2.2.2. Ultra-Wideband Based Positioning Systems

UWB-based localization has gained a lot of interest in research as well as widespread usage in industrial applications. Both classes will be briefly reviewed to provide an overview of the current status and promising applications of UWB localization.

2.2.2.1. UWB-Based Systems from Research

Mahfouz *et al.* present a surgical navigation system, which enables tracking of smart surgical tools and the logging of relevant objects such as a spacer block in the surgical scene [66]. An autonomous mobile security robot that is called CoLORbot is proposed in [67]. CoLORbot enables the environmental imaging of an unknown indoor area by using

UWB-Radar devices and a range point migration method. Furthermore, the calculated maps enable an autonomous navigation of the CoLORbot in an unknown environment. Fall *et al.* studied a localization system for railway transport that is based on UWB and the Time Reversal (TR) technique [68]. The TR technique supports an optimized signal detection by increasing the received energy. The combination of the TR and UWB technique allows a precise localization of the trains.

2.2.2.2. UWB-Based Systems from Industry

A commercial indoor localization system for assets and people tracking using UWB is offered by the Zebra Enterprise Solutions [69]. This is a Real-Time Locating System (RTLS) called Dart UWB operating in a range up to 200 m line of sight and reaching an accuracy up to 30 cm by using the TOA method. Figure 2.8 illustrates a UWB starter kit containing various components such as a Dart hub or a tag; whereby the Dart hub includes a location software to locate tags, which can be affixed to assets or worn by persons.

Figure 2.8.: *Starter kit from the Zebra Enterprise Solutions [69]*

Another player in the UWB commercial market is the Time Domain corporation that offers the UWB P440 transceivers, which can operate in a range up to 240 m [70]. The P440 modules use the two-way TOA method and have an accuracy of 2.1 cm (see Figure 2.9). The P440 transceivers allow the implementation, for example, of an RTLS in a hospital environment to track medical equipment, staff, or patients.

RTL-Service is a Russian company that offers a RTLS for localization and tracking of people and objects [72]. The RTLS is based on the UWB modules from the DecaWave company. The DecaWave modules are suitable for the TDOA and the TOA methods and enable an accuracy of about 10 cm and a data update frequency of 1-10 Hz. Furthermore, UWB modules have a range of up to 300 m [73].

2.2.3. Magnetic-Based Positioning Systems

Magnetic indoor localization systems can be classified into three categories: fingerprinting (geomagnetism), permanent magnet-based, and current-based magnetic positioning systems.

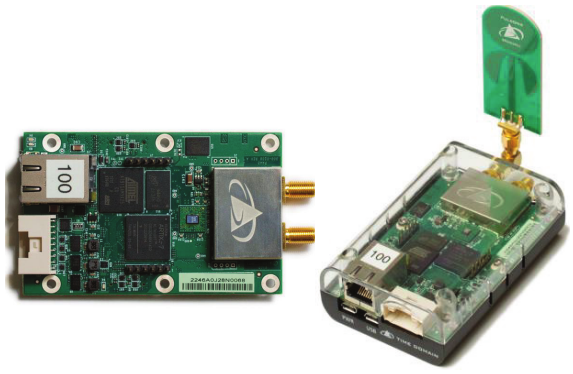


Figure 2.9.: *The PulsON 440 (P440) module from the Time Domain corporation [71]*

These systems have various advantages and disadvantages, which are listed in Table 2.2. Examples of each category will be briefly discussed in this section.

Table 2.2.: *Advantages and disadvantages of magnetic positioning (pos.) systems. NLoS, Non-Line-of-Sight*

Magnetic Pos. Systems	Advantages	Disadvantages
Fingerprinting system	No infrastructure.	Acquisition of magnetic maps in the setup phase. Localization accuracy scales with the fingerprinting data resolution.
Permanent magnet system	High accuracy. Operating under NLoS conditions.	Restricted coverage volume (up to $1\,m^3$). Mathematical models with high-order non-linear equations [49].
Current-based system	High accuracy. Operating under NLoS conditions. Not affected by reflections and multipath.	Infrastructure-based. Limited coverage area. Necessity of high power energy or highly sensitive magnetometers for coverage area extension.

2.2.3.1. Fingerprinting-Based System

The first class of magnetic localization systems uses magnetic field fluctuations inside buildings. Whereby, the source of the magnetic field fluctuations can be natural such as the earth's magnetic field, or man-made such as the electric power systems, industrial devices, or the steel and reinforced concrete structures [74]. Magnetic fingerprinting can be applied as a global self-localization if the anomalies enable a unique magnetic fingerprint and they are nearly static as well as locally available inside a building [74].

A fingerprinting positioning system is presented in [74] which is based on the anomalies

of the ambient magnetic fields. The system suggests that the ambient magnetic field may remain sufficiently stable over long time periods. Furthermore, the system can only be used for one-dimensional location cases, as for example, the localization of a person or a robot within corridors. No coils for the generation of magnetic field are needed; only a three-axis magnetometer is used to achieve an object or human self-localization. However, magnetic maps should be provided prior to the localization process. The magnetic maps are created for predefined pathways in the initialization phase. The proposed approach can be deployed parallel to other positioning techniques, such as a range finder or machine vision methods.

2.2.3.2. Permanent Magnet-Based Systems

The second type of magnetic positioning systems performs a localization based on magnetic fields created from permanent magnets. A positioning system that is based on permanent magnets can be composed of magnetometers as reference points and a permanent magnet as a mobile target. Alternatively, multiple permanent magnets with known locations can be used as reference points to locate a mobile magnetometer. Song *et al.* present a positioning system that is composed of a cylindrical permanent magnet [75]. The permanent magnet is enclosed in a capsule and incorporated into a human body, whose position is now determined by the magnetic signals measured from the magnetometer array. The system achieves an average position deviation of about 1.8 mm. Pham *et al.* propose a real-time magnetic tracking system that comprises a permanent magnet and magnetometers. Based on the measured magnetic signals, the tracking is calculated on a Personal Computer (PC) with an accuracy of about 5 mm [76].

2.2.3.3. Direct Current-Based Magnetic Field Signals

Magnetic fields can be generated by using direct or alternating current and permanent magnets. We focus on direct-based magnetic field signals in this section and we describe the current-based magnetic localization systems in the next Section 2.2.3.4.

Magnetism is a natural phenomenon that has been known since ancient times [77]. It is interesting to note that positioning was one of the first applications of magnetism. The compass provides, with respect to the earth's magnetic field, one of the earliest forms of navigation [78]. The compass complemented other navigation methods such as the use of the sun, stars or landmarks.

Jean-Baptiste Biot and Félix Savart describe the magnetic induction $\vec{B}$ caused by a wire carrying an electrical current I (see Figure 2.10). The infinitesimal magnetic induction $\mathrm{d}\vec{B}$ at point $\vec{r}$ can be calculated by applying the Biot-Savart law, as follows [79, 80]:

$$\mathrm{d}\vec{B} = \frac{\mu_0}{4\pi} \frac{I \mathrm{d}\vec{s} \times \vec{r}}{r^2}, \tag{2.11}$$

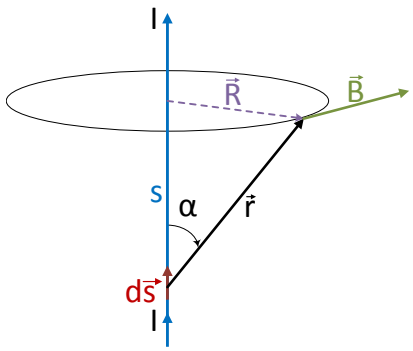


Figure 2.10.: *Biot-Savart law*

whereby μ_0 is a constant called the permeability of free space:

$$\mu_0 = 4\pi \times 10^{-7} T \frac{m}{A}. \tag{2.12}$$

The differential length vector $\mathrm{d}\vec{s}$ and the point vector $\vec{r}$ span a plane perpendicular to the vector $\mathrm{d}\vec{B}$, which means that the magnetic induction is given as a vector product. To calculate the entire magnetic field $\vec{B}$ created at some point, all the contributions from the current elements $I\mathrm{d}\vec{s}$ should be summed up. In other words, the total magnetic field $\vec{B}$ can be calculated by integrating Equation (2.11):

$$\vec{B} = \frac{\mu_0 I}{4\pi} \int \frac{\mathrm{d}\vec{s} \times \vec{r}}{r^2} \tag{2.13}$$

2.2.3.4. Current-Based Systems

The third category artificially generates magnetic signals by using coils (beacons) with known positions (reference points). The magnetic fields can be generated from coils by using pulsed Direct Current (DC) or Alternating Current (AC). Currently, commercial current-based magnetic positioning systems are designed for motion tracking and virtual reality in several artistic, industrial, and bio-medical applications, which require a small coverage volume (typically < 1.5 m^3) [81, 82]. In that context, three orthogonal concentric coils are used, and the magnetic sensor is connected to a central unit. The central unit activates the coils, collects the sensors' magnetic field data and performs the localization estimation.

Sheinker *et al.* propose a 3D experimental positioning system that is based on low frequency magnetic fields. The system is composed of coils (beacons) that are excited by an AC source to generate a time-varying magnetic field [83]. In addition, the MS includes a tri-axial search-coil magnetometer, six blocks of phase lock-in amplifiers and a location calculator. The system has an effective area of about 100 m^2 and can be deployed, e.g., for robot navigation and underground cavity mapping.

De Angelis *et al.* describe the design and implementation of an indoor positioning system,

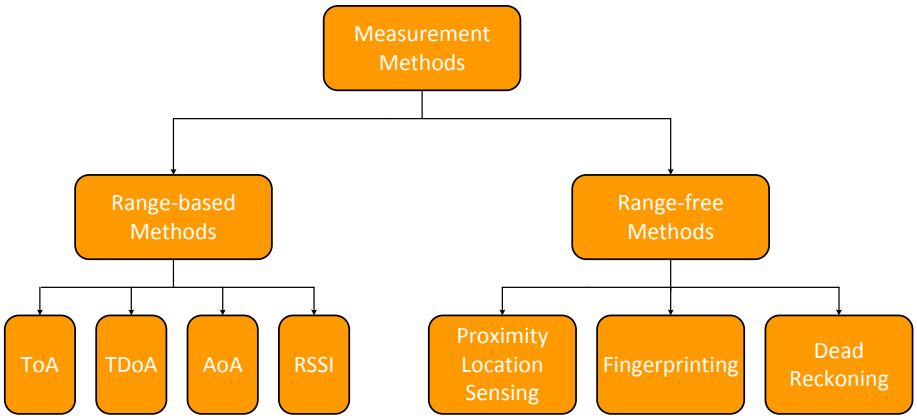


Figure 2.11.: *Measurement methods classification*

which is also based on AC magnetic fields [84, 85, 86]. The system consists of transmitter coils that are placed at known positions and a receiver coil. Each transmitter coil is driven by a signal generator to generate an electromagnetic field, which interacts with the receiver coil by inducing a current. The position of the receiver coil is estimated based on the Root Mean Square (RMS) voltage, which is measured at the receiver node. The system performs in two phases: the calibration and the trilateration phase. In the first phase, the coils and the receiver node are placed in line-of-sight conditions in an indoor environment to determine the calibration parameters. In the second phase, the position of the receiver node is calculated by using the trilateration method based on the received RMS voltages from the transmitter coils.

2.3. Conclusion

Indoor localization systems are required for a person or object tracking, for example, in healthcare institutions or logistics centers and warehouses. They enable personnel safety as well as the increase of assets productivity. These positioning systems are based on various measurement methods, which can be classified in range-based and range-free methods [87], as illustrated in Figure 2.11.

The comparison of these methods shows that there is a trade-off between the measurement error, complexity, and cost. Range-free methods such as fingerprinting can be achieved with simple, built-in sensors, for example, a WLAN-interface in a smartphone. In contrast, range-based methods such as TOA and TDOA can enable a reliable and accurate measurement. Nevertheless, they need additional and complex equipment such as UWB transceivers, tags, or precise synchronized reference points.

Localization systems can be classified into active and passive systems [5]. In the case of active localization, the object or the person participate actively by mounting or wearing an

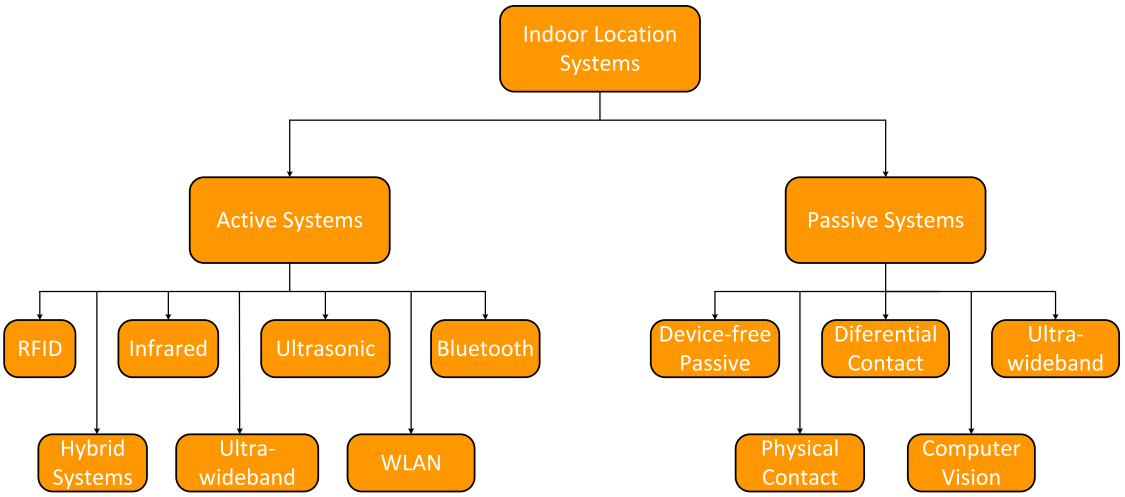


Figure 2.12.: *Classification of localization systems. Figure is adapted from [5]*

electronic device, which sends the position or recorded data to a positioning system or an application server, respectively. In the case of passive localization, the object or the person is not mounting or wearing any electronic devices to deduce their positions. The position is estimated by using the variance of a measured signal or video process [5]. Figure 2.12 outlines the active and passive localization systems.

Localization systems use various technologies such as WLAN, UWB, or RFID. The selection as well as the design of a localization system, depends on the granularity and accuracy required for a specific application and a deployment environment because there are currently no standards for indoor positioning systems.

CHAPTER 3

Operating Systems for Resource-Constrained Devices

In recent years, Operating Systems (OSs) for resource-constrained devices as well as for processor-based embedded systems have gained interest. These OSs are designed to operate on small devices including sensors and communication interfaces. Networks using these devices can be deployed in a wide range of disciplines and applications, which also include the following examples:

- **Efficient Power Supply for Data Centers**: The heat distribution of the network devices and computer servers consolidated in a data center is a key operational parameter that affects the energy consumption and the cooling behavior [88]. Energy savings have great social and economic impact for these centers as they consume huge amounts of energy (billions of kilowatts). Half of the energy is used by conservative cooling systems due to the over-cooling or over-provisioning with the power of devices. The deployment of wireless sensors across the machine enclosures allows for a reduction in power consumption by automatically sensing the areas that require cooling. The dynamic adaptation of the power consumption of the cooling system as well as of the devices make the Information Technology (IT) infrastructure more efficient and environmentally-friendly.

- **Monitoring of Active Volcanoes**: Wireless sensors allow the monitoring of unapproachable environments such as a volcano without the use of cumber telemetry systems. The wireless sensors build a network to record eruptions, earthquakes, or seismo-acoustic events by using sampled seismic and acoustic sensor data [89]. The sensor nodes locally compare the observed events to deduce their positions and to report aggregate data to a camp using a long-range wireless link.

- **Bridge Safety:** The Structural Health Monitoring (SHM) of civil infrastructures such as a bridge can be realized by using a WSN [90]. WSNs do not require expensive equipment and installation, and the high-cost of wired solutions. The SHM is implemented by using accelerometers, which gather synchronized data of the oscillations and the movement within the structure of the bridge. The collected data permit the

observation of the structural health of the bridge in response to events such as damage after an earthquake or high winds.

Although the previously discussed applications show only a small slice, where the networked sensors can be deployed, they show differences between these networks and other computing systems [91]:

1. They act in and respond to an unpredictable environment by collecting sensor data.
2. They use a wireless link due to the robustness, the deployment flexibility and maintenance simplicity, and the low cost.
3. They operate as a stand-alone.

Resource-limited devices such as a microcontroller-based sensor nodes play a key role in those applications. However, before we discuss various OSs for resource-limited devices, we provide a brief description of a microcontroller-based sensor node.

3.1. Resource-Constrained Devices

Currently, resource-constrained devices are deployed in the context of IoT, which is an evolution of the WSN [92]. The IoT enables the WSN to connect to the Internet and to move away from proprietary and closed systems to the Internet Protocol (IP)-based sensor networks [92]. Devices used in IoT are mostly low-cost, have limited energy resources, and are equipped with low-end processors and little memory. Typical resource-constrained devices are outlined in Table 3.1, which reveals their limited computation and memory resources compared to a PC. Furthermore, Figure 3.1 shows the OpenMote, TelosB Mica2, and the AVR Raven devices, respectively.

Table 3.1.: *An overview of various resource-constrained devices*

Device type	Vendor	MCU	Frequency	RAM	Flash
Discovery Board	ST Microelectronics	32 Bit STM32F4	168 MHz	192 KB	1024 KB
OpenMote	OpenMote Technologies	32 Bit CC2538SF53	32 MHz	32 KB	512 KB
TelosB Mote	Crossbow	16 Bit MSP430	8 MHz	10 KB	48 KB
TelosB Mica2	Crossbow	8 Bit ATMega 1284P	16 MHz	4 KB	128 KB
AVR Raven	Atmel	8 Bit ATMega 128L	20 MHz	16 KB	128 KB

(a) *OpenMote [93]* **(b)** *TelosB Mica2 [94]* **(c)** *AVR Raven [95]*

Figure 3.1.: *Examples of resource-constrained devices:*

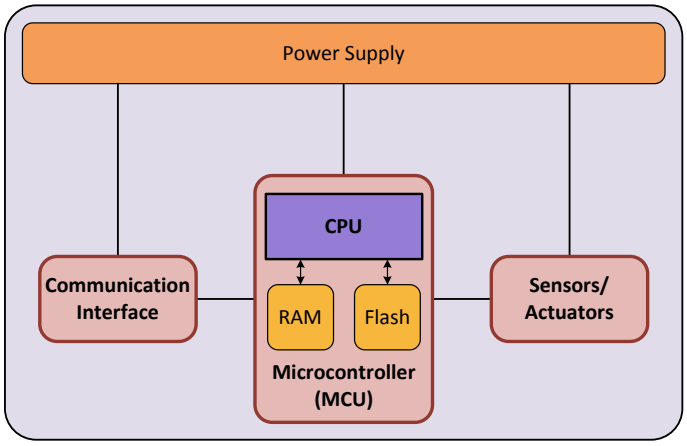


Figure 3.2.: *The main hardware components of a resource-constrained device*

As illustrated in Figure 3.2, a resource-constrained device comprises the following components:

Power supply provides the device with the energy. The device can include a microcontroller, sensors, or communication interfaces. The power supply is usually provided by primary batteries, whereby energy harvesting offers an alternative [96].

Microcontroller unit is a central part of a device facilitating the processing of data, the execution of a code sequence as well as the control of sensors. The MCU is comprised of a CPU, Random-Access Memory (RAM), and a flash memory (*cf.* Figure 3.2). The MCU includes additional modules such as non-volatile memory, General-Purpose Input/Output (GPIO) pins, and hardware interfaces. Examples of hardware interfaces are the Universal Asynchronous Receiver Transmitter (UART), the Serial Peripheral Interface (SPI), or the Analog to Digital Converter (ADC). The MCU cores are obtainable for the 8-bit, 16-bit, and the 32-bit architectures, whereby the RAM and flash have a size of hundreds of kilobytes (*cf.* Table 3.1).

Sensors and actuators sensors represent the interface to the real world and enable the observation of the physical environmental [33]. They transform a physical phenomenon

to electrical signals; therefore, they can be classified, depending on their outputs, as analog or digital sensors. There is a wide range of sensors measuring the environmental parameters such as sound, image, temperature, magnetic fields, pressure, distance or position. The actuators can control and interact with the environment, for example, by switching on/off a relay, which controls a light bulb, a motor or another object in the environment. Hence, the sensors allow for the representation and capture of the physical world in the digital world; while the actuators enable performing actions in the physical world triggered in the digital world [97].

Communication interface data exchange is needed to send and receive data between individual IoT objects by using a communication device; whereby wired communication can be used such as a Controller Area Network (CAN), a Process Field Bus (Profibus) or a Local Operating Network (LON). A more interesting case is the wireless communication, by which various transmission media can be utilized such as Radio Frequencies (RFs), ultrasound, electromagnetic waves, or optical communication. Commonly, a receiver and a transmitter are required, which convert a bit stream coming from the MCU to radio waves and vice versa [33]. Devices combining the transmission and receiving tasks are called transceivers, which include a circuitry required for both tasks. The circuity is composed of a modulator, demodulator, amplifiers, filters, mixers and so on [33]. Commonly deployed transceivers by IoT devices are off-the-shelf and available from the usual distributors. They are single-chip solutions such as the TR1000 family from Murata Electronics or the CC1000 and CC2420 family from Texas Instruments [33].

3.2. Operating Systems for Resource-Constrained Devices

The OS allows the management and sharing of resources as well as the development of multi-tasking applications in a computer system. A real-time OS (RTOS) has the ability to meet certain deadlines at the right time. Real-time capability does not necessarily mean fast, but it denotes the deterministic behavior or the predictability of the response time [98].

An OS can be essentially characterized by the following key design issues [12]:

Kernel structure

The kernel can follow a monolithic model, layered approach, or microkernel paradigm. The monolithic kernel is a simple way to implement an OS, which consists of two spaces: a user space (unprivileged mode) and a kernel space (privileged mode). A monolithic architecture is difficult to maintain and extend due the lack of modularity, particularly when the system exceeds certain size. A layered-based architecture is composed of various layers, that allow the designer to organize the kernel in a hierarchical way. The main disadvantage of a layered approach is the difficulty to

define the layers and maintain a boundary between the user space and the kernel. Unlike other architectures, the microkernel provides a minimum set of functionalities as well as a simple abstraction of the hardware. This is reached by implementing minimal OS services like thread management or Inter-Process Communication (IPC). The microkernel approach improves the reliability, if one service fails, as for example a device driver; however, this will not cause the entire system to crash. Furthermore, it facilitates the integration with third-party modules as well as an enhanced extensibility based on the message change property.

Scheduler

The scheduler is the OS part deciding which task to run next by using a scheduling algorithm. The scheduling strategy is closely coupled with the real time support, the depth of user interaction, and the supported task priorities.

Programming model

The programming model provides an abstract view of the hardware for the application developers by hiding the hardware complexity as well as enhancing the code portability and longevity. In addition, it defines the context in which the tasks are executed: Either all the tasks are executed in the same context and no memory space fragmentation is needed, or each task runs in its own thread and allocates its own stack. The programming model is tightly coupled with the range of available programming languages for the application development. Typical programming models are the event-driven and the multithreaded model [99]. The first model is suitable for resource-limited devices, but it is not commonly used by traditional application developers. In contrast, the second model is familiar to programmers, but it is unsuitable for resource-constrained devices. Therefore, various light-weight multithreading programming models are developed for resource-restricted devices [99].

There are a lot of OSs for resource-constrained devices, which vary in architecture, programming model, scheduling, memory management and protection, communication protocol, and real-time support [99]. Examples of these OSs are: FreeRTOS [100], TinyOS [101], Contiki [102], MANTIS [103], Nano-RK [104], LiteOS [105], and RIOT-OS [12], with their features summarized in Table 3.2.

3.2.1. FreeRTOS

FreeRTOS is a simple and user-friendly OS, which supports the most used architecture for embedded systems such as ARM, PIC or Zilog's Z80 [106]. It is written in C, has about 2,200 lines of code, and enables various services such as memory management, task management, or inter-task communication. The scheduler provides three kinds of scheduling methods: cooperative, preemptive, or prioritized scheduling policy. Although FreeRTOS

Table 3.2.: *Comparison of operating systems. FIFO, First-In, First-Out. TinyOS, TOS*

OS	Architecture	Real Time	Scheduling	Programming Model	Programming Language
FreeRTOS	Monolithic	Full Support	Round-robin preemptive and cooperative	Threads	C
TinyOS	Monolithic	No Support	FIFO	Primarily event-driven, support for TOS Threads	nesC
Contiki	Modular	Partial Support	Event-based	Protothreads and events	C with some constraints
MANTIS	Layered	No Support	priority-based Round-robin	Threads	C
Nano-RK	Monolithic	Full Support	Rate monotonic and harmonized scheduling	Threads	C
LiteOS	Modular	No Support	priority-based Round-robin	Threads and Events	LiteC++
RIOT-OS	*Microkernel*	*Full Support*	*Tickless, preemptive and scheduling with priorities*	*Threads*	*C and C++*

is convenient for the most lightweight microcontrollers, it does not support low power management features like the most real time operating systems. This is because energy-saving methods of modern MCUs are platform-dependent [107, 108], and the OSs use periodical timer interrupts, in order to manage the timers and the system time.

3.2.2. TinyOS

TinyOS is developed at the University of California (UC) in Berkeley and relies on a component-oriented architecture [109]. The scheduler, which implements a simple First-In, First-Out (FIFO) policy, provides a two-level scheduling hierarchy for events and tasks. The tasks are processed in a run-to-completion manner and they cannot be self-suspended or preempted. TinyOS implements an event-driven programming model, which consists of interrupts and tasks [110]. TinyOS is written in a C dialect called nesC and uses one stack. The execution of complex tasks increases the number of event handlers, and therefore the complexity of the control logic.

3.2.3. Contiki

Contiki follows a modular architecture as well as supports an event-driven model at the kernel level. The kernel incorporates a lightweight event scheduler, which provides synchronous and asynchronous events. Furthermore, it uses a single stack shared between protothreads. These protothreads solve the problem of blocking invocations, which is a common drawback for even-driven and non-preemptive kernels [111]. This enforces a state machine programming style that increases the programming complexity. Like TinyOS, Contiki is based on an event-driven programming model. Furthermore, it uses protothreads and a supplementary library to support preemptive multithreading.

3.2.4. MANTIS

MANTIS is a small footprint OS with a size of about 500 bytes. MANTIS is developed at the Colorado University and enables a prioritized threaded programming like the Portable Operating System Interface (POSIX) thread model [112]. The scheduler uses a round robin scheme for threads with the same priority, where higher priority threads are scheduled before lower-priority ones. Furthermore, MANTIS supports semaphores, mutual exclusion, as well as a low-level stack for the serial and radio communication. MANTIS does not provide a memory protection mechanism [99].

3.2.5. Nano-RK

Nano-RK is a real-time OS that supports both hard and soft real-time applications by using various real-time scheduling algorithms such as the rate harmonized or monotonic scheduling [99]. It uses a monolithic kernel design, which enables the use of a static design-time framework such as deadlines or task priorities to guarantee that deadlines are met. Nano-RK offers the programmer a familiar multitasking paradigm as well as provides a lightweight networking protocol stack that includes a communication abstraction which like the socket interface. It also provides many wireless link layer protocols such as RT-Link or B-MAC. In addition, Nano-RK supports two families of microcontrollers the Atmega and the MSP430x family.

3.2.6. LiteOS

LiteOS is a multithreaded OS that provides a Unix-like abstraction for resource-constrained devices [105]. It follows a modular architecture design. LiteOS enables a thread-based programming mode as well as event handling by using callback functions. In addition, it supports the object-oriented programming in the form of LiteC++ as well as provides a hierarchical file system and a Unix-like shell. The kernel implements both the round-robin and the priority-based scheduling. Furthermore, the kernel supports dynamic memory

Table 3.3.: *Measured RIOT-OS latencies of the LPC2387 MCU with a 72 MHz operating frequency; source: [107]*

Interrupt Types	Cycles	Time in µs
Interrupt latency	50	0.700
Context switch outside an ISR	72	0.990
Context switch	600	8.4
Inter-Process Communication (IPC) delay	1300	18

allocation, which enables the user to allocate and de-allocate memory at the run-time. LiteOS provides communication protocols at the Medium Access Control (MAC), network and transport layer, whereby detailed documentation about these protocols is not available [99].

3.2.7. RIOT-OS

The RIOT-OS [12] is an open source IoT operating system developed at the "Freie Universität Berlin". RIOT-OS is based on a microkernel architecture, which is inherited from FireKernel [107] deployed for a rescue scenario to track and monitor fire fighters. To fulfill real-time requirements for severe industrial or emergency scenarios, the micro-kernel provides a zero-latency interrupt handling and prioritized threads with a minimum context-switching time. Regardless of the system load, the maximum interrupt latency of the LPC2387 MCU operating at 72 MHz amounts to 50 cycles (700 ns). The maximum context switch time outside an Interrupt Service Routine (ISR) is 72 cycles (900 ns) (*cf.* Table 3.3).

A tickless scheduler is implemented to achieve a maximum energy savings and to support a deep-sleep mode by all resource-constrained MCUs. Since most schedulers wake up periodically to enable the switching between tasks, their behavior is dependent on periodic timers, which is not desirable for the energy awareness. Furthermore, the most constrained devices cannot be woken up from a timer interrupt, but only from external interrupt sources. In this manner, the periodical timer interrupts prevent the deep-sleep mode leading to excessive power consumption over the entire run-time. In addition, RIOT-OS features a developer-friendly programming model by supporting the standard C/C++ programming languages of the American National Standards Institute (ANSI). Furthermore, RIOT-OS features standard multithreading, real-time behavior, and POSIX-like Application Programming Interface (API) for all supported hardware, which scales from 16-bit microcontrollers to 32-bit processors.

3.3. Conclusion

This chapter has shown the necessary hardware components for resource-constrained devices, which can be embedded into daily objects and networked to build the IoT. These devices

are characterized by scarce energy resources, limited storage capacity, and low computation power. Therefore, the choice of components such as the MCU or the communication interface is application-dependent and a trade-off between various criteria and constraints such as the computation and the energy capacity. The use of an OS, which is a software part, provides the following benefits to applications: simplification of design, hiding hardware complexity, and increasing portability [113]. OSs provide various kernel architectures, scheduling algorithms, and programming models. The choice of the OS could be challenging, since IoT devices are resource-constrained and application-specific. The selection criteria of the OS can be real-time capability, energy awareness, standard APIs support, or network protocols.

CHAPTER 4

Open Platform for Positioning Systems

A platform architecture for positioning systems is essential for the realization of a flexible localization system, which interacts with other systems and supports various positioning technologies and algorithms. The decentralized processing of a position enables to push the application-level knowledge into a mobile station and avoids communication with a central unit such as a server or a base station. In addition, the calculation of the position on low-cost and resource-constrained devices presents a challenge due to the limited computing, storage capacity as well as the power supply. Therefore, we propose a platform architecture that enables the design of a system with the reusability of the components, extensibility (e.g., with other positioning technologies) and interoperability. Furthermore, the position is computed on-the-fly on a low-cost device such as a microcontroller, which simultaneously performs additional tasks such as data collection or preprocessing based on an operating system. The platform architecture is designed, implemented, and evaluated based on two positioning systems: time-of-arrival and the field strength-based localization system. The platform architecture is open in terms of being able to interact and communicate with other systems as well as an open source. The platform architecture enables a decentralized, continuous as well as an accurate localization [14].

4.1. Architectural Overview of Localization Systems

Positioning systems can be classified into three categories: centralized, decentralized, and distributed architecture. Figure 4.1 illustrates different architectures. The architecture plays a major role in the performance, reliability and energy consumption of a positioning system. We distinguish between the architecture of the whole positioning and the components of the system, such as the MS: The positioning system architecture describes the interaction and the coordination between all system components, whereby a component can be a mobile or a base station. The second specifies the design and implementation of an MS or a base station. As data processing plays a key role in a positioning system, we firstly compare different architecture approaches in this section.

The processing of the data or the location can be performed by using three classes of architectures for positioning systems: central, decentralized and distributed.

- Centralized Architecture: This is the most commonly-used architecture, in which the

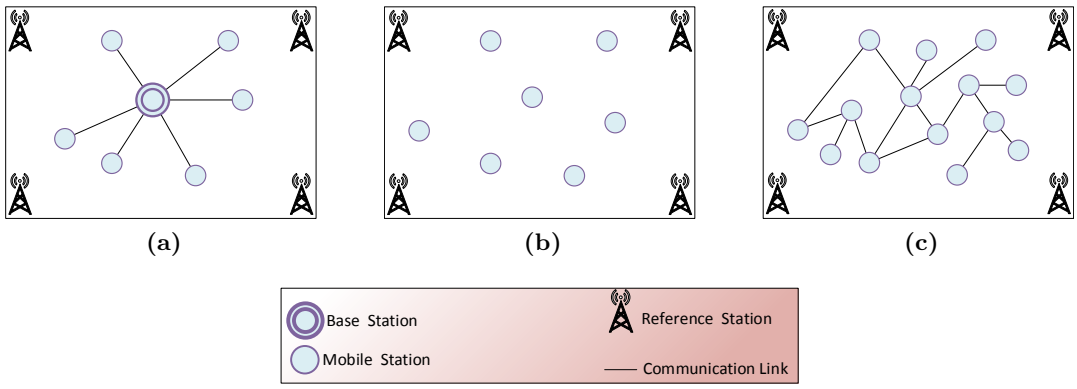


Figure 4.1.: *Position evaluation: (**a**) central, (**b**) decentral, and (**c**) distributed*

sensors or the MSs exclusively communicate with the base station. A base station usually possesses more processing power, storage capacity and energy resources than the MSs. The MS modules deliver data to the base station, such as raw sensor data or the results of a signal processing step; normally, with the compression or reduction of the data. The individual MSs have no knowledge about the semantics of the gathered data that is transmitted to and interpreted by the base station. The advantages of a centralized architecture include the usage of lightweight and low-cost MSs, since the whole complexity is shifted to the base station; a high position resolution and a full central overview of the observed phenomenon (e.g., event). The disadvantages comprise: a single point of failure (e.g., the base station failure), which is not tolerable in a secure scenario, and poor performance with a large number of MSs. In addition, the system can fall into an energy starvation in case of continuous communication between the MSs and the base station, whereby the power-saving techniques are difficult to implement.

- Decentralized and Distributed Architecture: In this architecture, data processing and position computation occur in the mobile station, and no data, except the result of a position finding, are sent to the base station. Furthermore, in a distributed MS network, which is also referred to as collective evaluation [114], the MSs can exchange data to collectively achieve and respectively augment the efficiency and the precision of a positioning task [115]. This can be important in the case of using distributed localization algorithms [116]. The distributed approach also favors cooperative sensing and positioning when multiple MSs are present. The decentralized and the distributed architecture exhibit the following advantages:

  1. Scalability: the computational load and the communication overhead at each MS do not depend on the MS number.
  2. Robustness: the system is not affected if an MS or some MSs fail, since no node

has a designated role and all nodes can perform data processing and transfer tasks.

3. Energy-awareness: the data traffic between the base station and the MSs is reduced, since this architecture supports on-board or in-network processing, saving both link bandwidth and node energy, which are critically-constrained resources [117].

The drawbacks of the decentralized and distributed architectures are: The MSs have no global knowledge about the network topology or about a phenomenon. Components of the distributed architecture only know about connections in their own neighborhood, which requires a robust and self-healing network. Since the aim of the decentralized or the distributed architecture is on the mobile station or in-network processing, the use of lightweight and low-cost MSs is not possible. The MS should have an MCU with more computational power and memory capacity, as well as an efficient application layer and, in the case of a distributed evaluation, a communication protocol stack. The distributed position evaluation is not the focus of this work. We use the decentralized architecture due to the scalability, robustness and energy-awareness and on-the-fly capability of the MS.

4.2. State of the Art

Firstly, we compare various Indoor Localization Systems (ILSs) from the industry as well as from the research area in terms of their platforms and architecture. Finally, we give a brief overview of various standards for localization systems.

4.2.1. Centralized Indoor Localization Systems

Ubisense has developed a RTLS based on the UWB radio technology, which allows simultaneous tracking of many tags by combining the AOA and the TDOA methods [118]. The position of the tags is calculated in a location server using the signals transmitted from the tags to the reference points synchronized by timing cable. The Ubisense RTLS has a centralized architecture which can be divided in three parts: the sensor network, the location engine, and the location platform. The sensor network comprises the tags and the reference point sensors, which are distributed over the area and receive the UWB pulses form the active tags. The location engine, running on a server, presents the software platform. It enables the position calculation based on the data received from the reference points as well as setting up the sensor network. Moreover, the location platform allows the user to define location-based services and the visualization of the target area, for example, the highlighting of an object entering a defined zone.

The Ekahau RTLS is a fingerprinting location system based on the RSSI method using an existing WLAN-network to enable the tracking of phones, bar code scanners, or people wearing WLAN tags [119]. Radio maps should be created in an initialization phase based on

the RSSI and the coverage information of each Access Point (AP). The localization system is centralized and includes a positioning engine, a site survey, and a client. The positioning engine is a server that computes the position of the tag by using calibration information and a probabilistic location detection algorithm. The site survey collects the coverage and RSSI information of each AP. Furthermore, the client runs on a device such as a phone or WLAN tag.

Wang *et al.* have developed a high precision indoor localization system, which is centralized and based on UWB technology [120]. This centralized architecture is achieved by using a gateway and a control server. The mobile stations that are connected to the UWB transceivers via Bluetooth transmit the measured distances between the mobile and the reference stations to the gateway using a WLAN interface. The gateway supports up to ten simultaneous connections with the MSs to forward the measurement data frames to the control server. This, in turn, computes the position of the MS by using the trilateration algorithm.

4.2.2. Decentralized Indoor Localization Systems

The most decentralized localization systems are based on smartphones, which use the on-board sensors to calculate the position of a user. Smartphones are significant information interfaces between the user and the environment possessing a substantial computational capacity and communication capabilities [121]. They allow for the use of built-in sensors such as an accelerometer, magnetometer, camera, communication interfaces as well as the use of existing infrastructure such as the Internet or WLAN APs. Nonetheless, smartphones incorporate relatively low-cost sensors, which do not facilitate an accurate localization with centimeter- or millimeter-level accuracy. In this work, we want to achieve a cm-level accuracy, which allows pedestrians and robots to navigate inside a building. Such an accurate localization cannot be obtained from sensors embedded in smartphones. Furthermore, potential energy consumers are the screen, continuous usage of on-board sensors, and the communication interfaces. Smartphone-based localization systems can be classified in signal- and inertial-based mobile ILS [122].

4.2.2.1. Signal-Based Indoor Localization Systems

Zhuang *et al.* developed an indoor localization system enabling simultaneous map acquisition and repeated tracking (SMART) [123]. The indoor positioning and the map construction of the unknown environment are achieved by measuring the radio signals of the WLAN APs as well as sensing the motion speed and heading of the mobile device. The motion speed and heading are measured by the built-in accelerometer and magnetometer, respectively. The localization module is implemented on a smartphone to achieve a localization estimation, while the ambient fingerprinting and the map construction is performed on a server.

ARIEL proposed an automatic WLAN-based localization, which enables fingerprinting of rooms in a building without the need of a floor plan [124]. The built-in accelerometer is used to improve the accuracy of the system by correlating motion patterns with the signal strength of the WLAN. ARIEL enables the room localization of a person. The smartphone collects signal vectors, which are extended with the motion data from the accelerometer, and delivers them to the server. The smartphone caches a local database storing the fingerprints of the rooms that the user already visited to locally perform a room localization without the deployment of the server. If a person enters an unvisited room, the smartphone must perform WLAN-scans to acquire localization services. The scanning time is approximately thirty minutes, when a user enters an unvisited room whose fingerprint has not been established.

Although the signal-based smartphone ILSs allow for the localization of users by using existing infrastructure, most of them are based on fingerprinting, which needs a calibration phase by manually collecting a vast amount of training data [125]. In addition, a retraining process is necessary if the deployment environment is altered. Signal-based smartphone ILSs do not enable continuous (smooth) localization. Furthermore, they show limited accuracy due the instability and unreliability of the RSS and the absence of a causal relationship between the Euclidean distance and the RSSI [126]. A higher accuracy can also be achieved by using special hardware, not readily available for smartphones; or by using hundreds of APs which are not practical [127].

4.2.2.2. Inertial-Based Indoor Localization Systems

Park *et al.* present a pedestrian tracking system that uses a smartphone incorporating a magnetometer and an accelerometer sensor [128]. This tracking system is deployed for indoor corridor environments, whereby the pedestrian location is estimated by using sensor data and a Hidden Markov Model (HMM) determined by a Bayesian filter. The system is implemented for various smartphone types as well as evaluated for four possible positions of the device, for example, the user calling or pocket positions.

A pedestrian dead reckoning for indoor localization (SmartPDR) is developed by Kang and Hal using inertial sensors embedded in the smartphone [129]. SmartPDR recognizes the step event and computes the step length by using a three-axis accelerometer. The heading direction is calculated by means of the magnetometer.

Smartphone inertial-based ILSs enable the localization of a user by using the on-board magnetometer and accelerometer allowing for a continuous localization for only a certain time due to the drift error of the inertial sensors. In addition, compass fluctuations due to variable magnetic field induced by the indoor environment (e.g., ferromagnetic building materials) lead to inaccurate heading estimations. Therefore, complicated algorithms are needed to cope with these errors [130]. A starting point is essential for the tracking, which requires the use of an external positioning system or the intervention of the user. Furthermore, a

calibration process is needed for the estimation of the stride length of the user [122]. The accuracy of the localization depends on the smartphone orientation and position with respect to the user's body [122].

4.2.2.3. Other Decentralized Indoor Localization Systems

Besides smartphones, there are other platforms that compute the position locally: Tango is a platform using computer vision, inertial tracking, depth sensing and machine learning to perform a position calculation [131]. Although Tango achieves centimeter-level indoor positioning, the lighting conditions as well as scene similarity remain a challenge for Tango devices. Thus, the positioning accuracy may decrease to meter-level [132]. Open Shoe is an open source project for creating an embedded foot-mounted Inertial Navigation System (INS), which enables pedestrian positioning and consists of an Inertial Measurement Unit (IMU) and MCU [133]. The INSs also suffer from the problems of inertial sensors, although sensor fusion methods such as Zero Velocity Update (ZUPT) can slightly correct the drift but cannot counteract the effect of the forces during running. Many intelligent robots are based on autonomous navigation and Simultaneous Localization and Mapping (SLAM) for creating maps, whereas the localization plays an important role. Endo *et al.* show the application of SLAM for visually impaired navigation systems based on Large-Scale Direct (LSD)-SLAM [134]. SLAM-based systems allow accurate real-time position over long distances without the drift as well as running algorithms locally on a robot [135]. Nevertheless, challenges for SLAM algorithms are the map convergence as well as the computational and memory requirements for real-time/real-world implementation [136].

4.2.3. Distributed and Cooperative Localization Systems

Schmid *et al.* present a proof-of-concept of an ad-hoc localization system for persons in a WSN [137]. The initial positions of the anchor nodes are determined by the Pedestrian Dead Reckoning (PDR) technique. In contrast, the approximate positions of the mobile nodes (on-body nodes) are estimated based on the positions of the anchors, which broadcast their positions in regular time intervals. The sensor nodes gather only sensor data by using a ZigBee stack protocol allowing a multi-hop communication between the sensor nodes, whereby the algorithms are performed off-line.

Yamagushi *et al.* present two approaches for collaborative indoor localization, which are the stop-and-go localization and the People-Centric Navigation (PCN) methods [55]. The first approach relies on the use of ranging devices, whereas a stop-and-go activity is performed to select a suitable set of reference nodes. The TDOA technique is used for distance measurement, which is supported by ultrasound devices. Each sensor node incorporates various ultrasound transmitters radially arranged to reach an omni-directional range pattern. The second approach is based on collaborative PDR, whereby each mobile

node provides its own trajectory to other nodes to form the group-averaged trajectory. The users estimate their step vector by using smartphones, enabling the estimation of the step count and direction.

Although, cooperative ILSs permit the improvement of the location coverage as well as the location accuracy, especially in the case of poor geometric conditions [11]. These systems must overcome operational requirements as well as technical challenges [55]. The operational requirements can be related to privacy protection, for example, users do not allow sharing of their positions; or the incentive of users to share their position, for example, by getting network services [138]. Technical challenges include the selection of the reference nodes, increasing energy efficiency which requires the optimization of space and time of localization, and the self-localization by mobile nodes [55]. Another technical challenge is the self-organization due the large numbers of MSs with random environmental characteristics. The error propagation is a serious problem in the distributed evaluation. In addition, most multihop localization techniques by cooperative localization in WSNs are not implemented and are only treated at the theory level; or they were tested in simulated environments [139].

4.2.4. Standards for Localization

Although there are several positioning systems from the commercial or research fields, the ANSI 371.1 RTLS and the Institute of Electrical and Electronics Engineers (I Triple-E) (IEEE) 802.15.4.x localization standards give only the specification of the physical and the MAC-Layer.

The ANSI 371.1 RTLS standard specifies the positioning accuracy as well as the physical layer of a real time location system called WhereNet and developed by Zebra Technology Company [140]. WhereNet supports both indoor and outdoor real-time positioning by using TOA method and the 2.4 GHz Direct Sequence Spread Spectrum (DSSS) technique [22, 25]. WhereNet is composed of tags sending signals at regular intervals to the time-synchronized base stations.

The IEEE 802.15.4a is the first international standard specifying the wireless physical layer to enable precision ranging [141]. Furthermore, it supports a prolonged range, high data rates, improved robustness against interference to allow for applications in wireless personal area networks, such as location-based routing and the tracking of moving objects. The physical layer is based on UWB technology and supports Impulse Radio (IR) UWB and Chirp Spread Spectrum (CSS) [40]. The MAC-Layer is specified at the IEEE 802.15.4-2003 and supports slotted and unslotted Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) [142]. The IEEE 802.15.4d-2009 enables the CSMA/CA mechanism specified at the original IEEE 802.15.4 standard [142].

4.3. General Architecture of a Platform for Positioning Systems

The design of a positioning system is an interdisciplinary challenge that combines approaches from various areas such as computer science, engineering, and mathematics. Therefore, the design and implementation of a positioning system is a complex process that requires skills in these fields.

The architecture of a platform includes data processing and the system component interaction and design. In the case of a centralized or the decentralized and distributed approach, data processing can be accomplished on a server or in a Mobile Station (MS), respectively. The system component can be an MS or an anchor. We use the decentralized architecture due to the on-the-fly capability, robustness, scalability, and energy-awareness of the MS.

As mentioned in Section 1.1, the platform should be open in the sense of the interoperability and communication with other open systems as well as open source. Furthermore, the platform should allow for decentralized, accurate and smooth position locating.

4.3.1. System Interaction and Components

The proposed anchor-based ILS is composed of reference stations (anchors) and an MS, which includes sensors enabling a distance or signal strength measurement. As illustrated in Figure 4.2, the MS performs a ranging or signal strength measurement to the reference stations, whereby the data processing as well as the position computation occurs on the MS. The localization process consists of three distinct phases: measurement, preprocessing and position estimation. In the first phase, the MS collects data after performing a measurement to the anchors. The measurement data are preprocessed in the second phase, for example, to eliminate outliers. Finally, the position is calculated at the third phase.

4.3.1.1. Layers Interaction of the Mobile Station

The suggested platform follows a modular-based architecture that ensures the portability and the extensibility of the system. As illustrated in Figure 4.3, the system architecture of the MS is divided into two layers: the system and the application layer.

The measurement data, such as distances or magnetic field strengths, are gathered as well as synchronized by the SL after performing measurements to the RSs. These measurements and their synchronizations are supported by the OS that controls the hardware sublayer including devices such as UWB-transceivers or magnetometers. The collected distances are delivered to the preprocessing sublayer for the removal of outliers or the calibration of measurement data. Finally, the location is calculated by the position-estimation sublayer using the data delivered by the preprocessing sublayer.

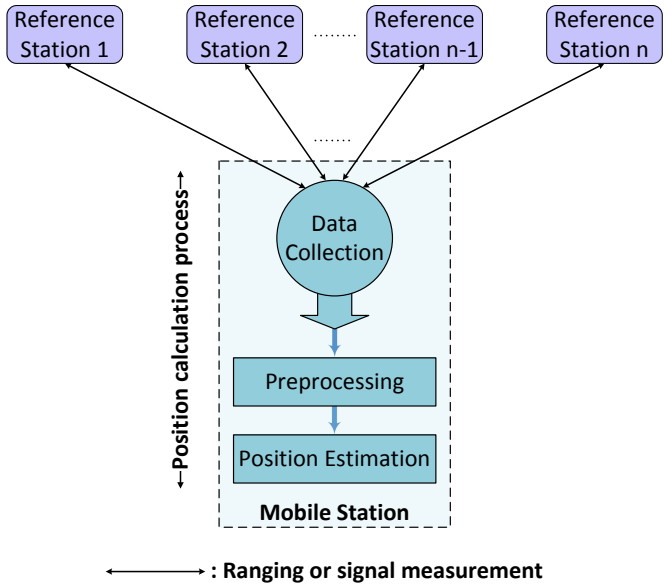


Figure 4.2.: *Components interaction of a localization system*

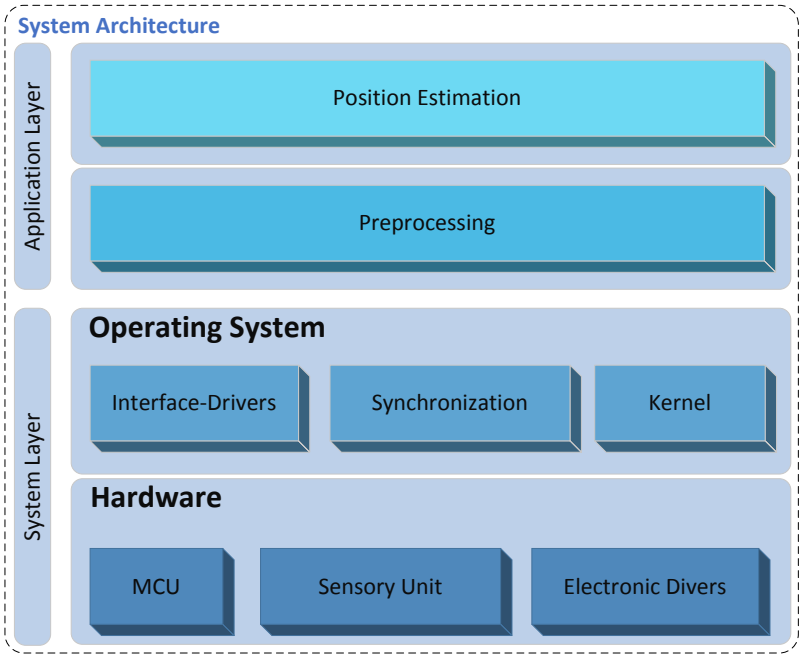


Figure 4.3.: *The architecture of the mobile station*

4.3.2. System Layer of the Mobile Station

The SL consists of two sublayers: the hardware and the OS. Both sublayers will be discussed in Sections 4.3.2.1 and 4.3.2.2.

4.3.2.1. Hardware Layer

The hardware sublayer includes a computing unit (e.g., MCU), sensory unit and electronic drivers. The sensory unit includes sensors such as UWB or magnetometer sensors to accomplish distance or magnetic field strength measurements. The electronic driver circuits enable, for example, interfacing with external driver units.

4.3.2.2. Operating System Layer

The OS simplifies the software design, hides the hardware complexity, and increases the software portability (*cf.* Section 3.2). The OS sublayer includes a software interface driver module, which provides interfaces to communicate with sensors or other devices. Most digital sensors support standardized interfaces such as UART, SPI, and Inter Integrated Circuit (I2C) buses encouraging the interoperability and the extensibility of the system with other technologies. A synchronization unit can serve to synchronize the received sensor data between a transmitter and receiver. Very precise clocks are required for some indoor localization techniques such as TOA or TDOA. The kernel is the innermost portion of the OS as well as the software part that provides basic services, distributes system resources, and manages hardware [143]. The principal components of the kernel are a scheduler, interrupt handlers, a memory management and protection mechanism, and system services such as IPC or networking [99].

4.3.3. Application Layer

To ensure the portability and the extensibility of the system with various applications and positioning algorithms, the AL as the highest level of the proposed platform, follows a modular-based architecture. The interoperability of the AL, as well as that of the MS can be extended by using an open-standard format such as the JavaScript Object Notation (JSON) [144] in order to enable data exchange with other devices. An example would be the data exchange between the MS and applications located on a PC. The AL is subdivided into two sublayers: the preprocessing and the position computing sublayers (*cf.* Figure 4.3). Both sublayers of the AL will be discussed in Sections 4.3.3.1 and 4.3.3.2.

4.3.3.1. Preprocessing

The preprocessing sublayer comprises data filtering that reduces the effect of statistical outliers from data delivered by the SL. The outliers can be filtered out by using the mean, the median or the Median Absolute Deviation (MAD) filters [145, 146]. The preprocessing sublayer can also provide a calibration routine to correct uncertainties in the measured data.

4.3.3.2. Position Estimation

The top-level sublayer provides the algorithmic core that computes the position. Commonly, a positioning algorithm is applied to estimate the unknown position of a MS based on measurements to reference points. The positioning algorithms must satisfy some practical requirements to be implemented in a practicable system. For example, the algorithm should be robust against noisy measurements, otherwise the performance of the algorithm can drastically decrease. Before we describe some possible algorithms, we briefly introduce several measurement methods along with performance metrics of positioning algorithms.

a) Measurement methods

The common measurements methods are based on signal strength, angular or distance observations such as RSS, AOA or TOA. The TOA is the most popular measurement technique, which can be estimated by using various ranging techniques such as the one-way or two-way TOA and TDOA. Additionally, hybrid measurements can be used for the positioning, for instance TDOA/AOA or TOA/RSS. See Section 2.1.

b) Performance metrics

The performance of a positioning algorithm can be evaluated by the following metrics: accuracy, precision, complexity, robustness, scalability, resilience to error and noise, coverage, and cost [10].

- *Accuracy metrics*: The localization accuracy metric shows how well the ground truth and estimated positions match. There are several accuracy metrics such as the Root Mean Square Error (RMSE), the Cumulative Distribution Function (CDF), the Probability Density Function (PDF) or the Frobenius metric (FROB) [147].

  Root mean square error is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{x}_i - x_i)^2}{n}}, \tag{4.1}$$

  whereas $\hat{x}_i$ and x_i are the estimated and the true values, respectively, and n is the number of observations.

  Discrete/Continuous Probability density function gives the probability density for discrete or continuous random variables [148].

  Cumulative distribution function $F(x)$ is the probability that a random variable X assumes a value less than or equal to a given x [149].

Frobenius gives the residual error between all n nodes in a network, which consists of the reference stations RS_i having a priori location information and unlocalized mobile stations (MSs). Figure 4.4 illustrates a network comprised of a MS and four RSs. The FROB estimates the RMSE of the total residual error, which outlines the global quality of a positioning algorithm [150].

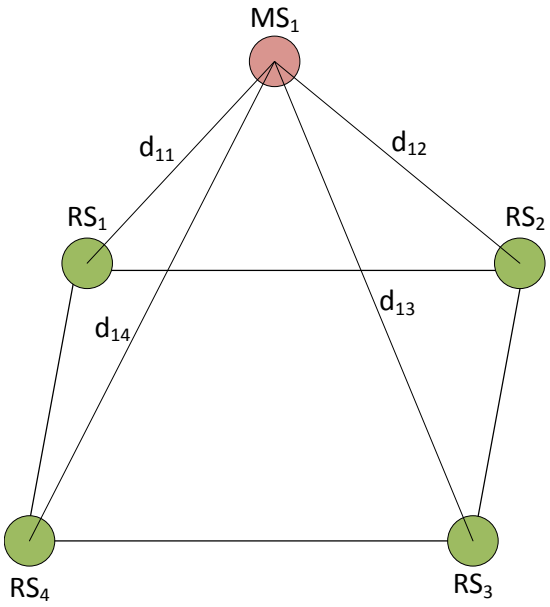


Figure 4.4.: *Example of a network*

The FROB can be calculated as follows:

$$FROB = \sqrt{\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\hat{d}_{ij} - d_{ij}\right)^2}\,, \tag{4.2}$$

whereby, $\hat{d}_{ij}$ and d_{ij} are the measured and ground truth distances, respectively.

- *Precision*: the location precision is defined as "the percentage of the results satisfying a predefined accuracy requirement" [151].
- *Complexity*: is determined in time and space and is commonly expressed by the standard notation big O ($\mathcal{O}$). The derivation of the analytic complexity formula of various localization algorithms is difficult to establish, therefore the computing time can be estimated [10]. The computing time depends on the method of processing the localization. The positioning algorithms could be quickly performed on a centralized server that possesses highly effective processing capability and sufficient energy supply. In contrast, the position can be carried out on a decentralized and mobile unit with constrained resources. A lack of computing and energy resources will affect the complexity [10].

- *Scalability*: is the ability of the algorithm to allow for the extension of the coverage area as well as the increase of the number of components such as the MSs or the reference stations.
- *Resilience to error and noise*: refers to the algorithm behavior in the presence of errors and noise in input data [150].
- *Cost metrics*: A practical evaluation criterion is the cost of an algorithm, which often is a trade-off against accuracy. Commonly cost metrics are the algorithm complexity, convergence time, power consumption, and the reference to node ratio [152].

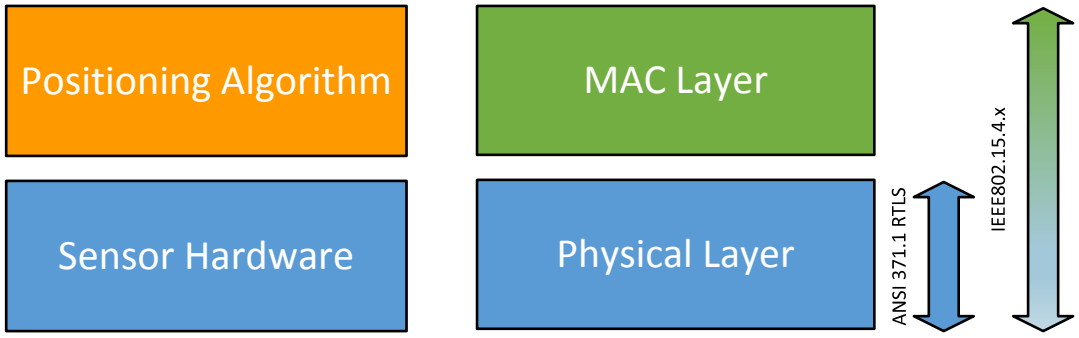
4.3.3.3. Positioning Algorithms

Positioning algorithms can be classified into two groups: Deterministic and probabilistic methods [11]. Deterministic methods determine directly the position based on the measurements by applying, for example, the lateration or the least squares method. The lateration is a popular location algorithm that computes the position of an unknown MS by measuring its distance from multiple reference positions. The algorithm is called trilateration if the number of reference points is three. Otherwise, it is called multilateration. On the other hand, probabilistic methods [153, 154], which are known as Bayesian methods, enable the position finding by considering the uncertainty of the measurements [11]. The Bayesian methods proceed in two steps: the prediction and correction steps.

4.4. Conclusion

Although many research and commercial location sensing systems have been developed by industrial and academic researchers, they are only roughly described. The architecture of most systems is divided in two parts: the sensor hardware and the positioning algorithm [10]. The standards for localization such as the ANSI 371.1 RTLS and the IEEE 802.15.4.x cover only the physical and the MAC layers. Figure 4.5 illustrates this relationship.

A modular-based architecture ensures the reusability of the software components as well as the portability and the extensibility of the system. The decentralized architecture enables to push the application-level knowledge into a mobile station and avoid the communication with a server or a base station. Furthermore, the use of an OS supporting standard protocols such as the Internet Protocol version 4 (IPv4) or 6LoWPAN allows the localization system to communicate with existing systems and protocols. It also allows the implementation of non-proprietary and open systems, which can be a part of the IoT. Therefore, the proposed open platform can be a guideline for the designer of a positioning system to achieve a system design with a high extensibility (e.g., with other positioning technologies), reusability of the components, and interoperability.



(a) *the industrial and academic area.* **(b)** *the ANSI 371.1 RTLS and IEEE 802.15.4.x.*

Figure 4.5.: *Localization system architectures:*

The Open-Source Mathematical Library for Resource-Constrained Devices (RcdMathLib) facilitates performing numerical linear algebra, positioning algorithms, and signal processing on resource-constrained devices (see Appendix A). It also allows the users a worldwide free use of the library as well as the modification of the source code for their purposes. In addition, the open source enables the developers to review and enhance existing source code [155, 156]. This library could enrich the RIOT-OS community with additional collaboration and innovation in regard to localization techniques.

CHAPTER 5

Decentralized UWB-Based Indoor Localization System

Based on the general discussion of the open platform discussed in Chapter 4, we now present a TOA-based system that makes use of the UWB technology. We apply the decentralized architecture for this positioning system [14]. The proposed system is composed of several UWB transceivers with known positions as reference stations and one MS, which also incorporates a UWB transceiver (see Figure 5.1). The UWB transceiver computes a peer-to-peer distance to the reference stations using the two-way TOA measurement technique [10]. The position is calculated on the MS based on the measured distances to the RSs.

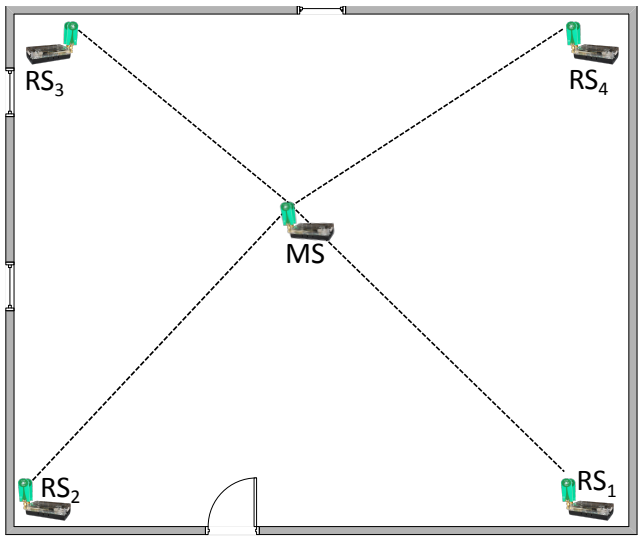


Figure 5.1.: *Principle of a UWB-based ILS. MS, Mobile Station. RS_i, Reference Station i*

Figure 5.2 presents the architecture of the MS. In the first phase, the distances are collected from the SL after performing distance measurements to the anchors. These measurements are supported by the RIOT-OS that controls the hardware including the UWB-transceivers. The collected distances are delivered to the preprocessing sublayer for the removal of outliers in the second phase. Finally, the position is calculated based on the data delivered form the

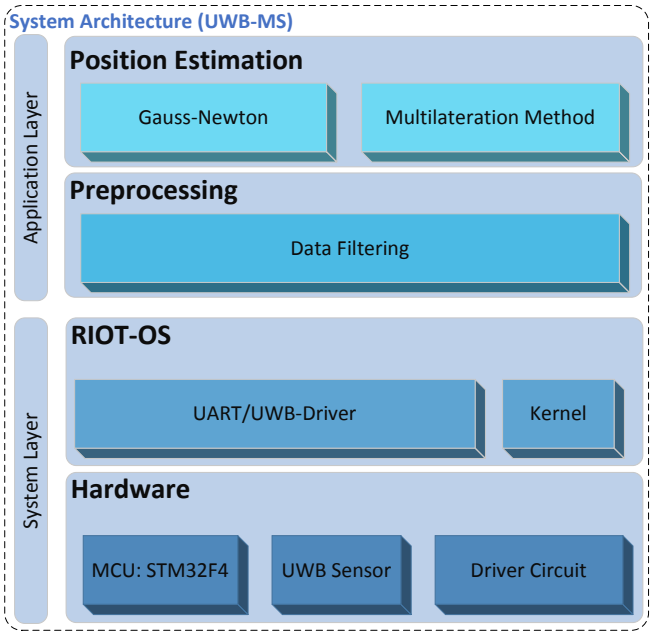


Figure 5.2.: *System architecture of a Time-of-Arrival based MS*

preprocessing sublayer in the third phase. The system as well as the application layer of the UWB-based positioning system will be described in Sections 5.1 and 5.2, respectively. Section 5.3 presents the complexity of the algorithms used. The system will be evaluated in Section 5.4 as well as later in Section 7.6. The chapter is concluded in Section 5.5.

5.1. System Layer of UWB-Based Mobile Station

As illustrated in Figure 5.2, the SL is based on the previously discussed architecture template (see Figure 4.3) and is composed of a hardware and an operating system sublayer, which will be described in this section.

5.1.1. Hardware

The hardware is composed of four subsystems: the power unit supplying the sensor board with energy, the MCU, the sensory unit and the driver circuits (see Figure 5.3). The hardware layer is implemented based on the STM32F407, ARM Cortex-M4 core operating at 168 MHz and the UWB module P440 ranging sensor from TIME DOMAIN®, which enables ranging measurements with an accuracy of a few centimeters. The properties of the MCU and the UWB ranging sensor used are summarized in the Tables 5.1 and 5.2, respectively.

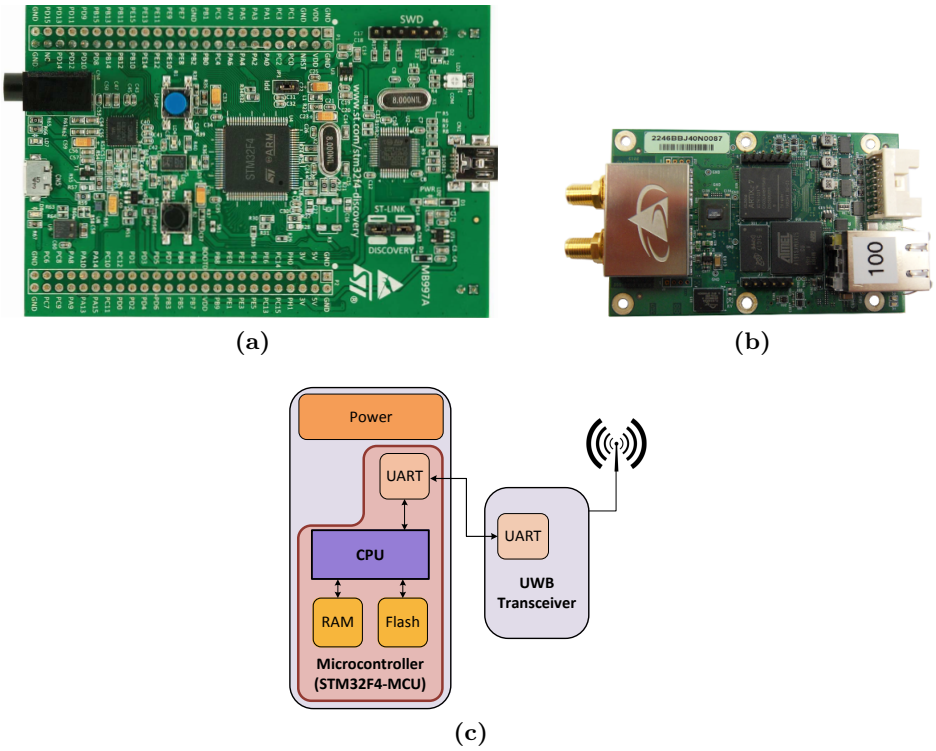


Figure 5.3.: *Hardware of the UWB-based localization system. (**a**) STM32F407 Discovery Board [157], (**b**) UWB P440 transceiver [158], (**c**) simplified hardware block diagram*

Table 5.1.: *The properties of the STM32F407 MCU. MCU, Microcontroller Unit [159]*

MCU	Family	Vendor	Frequency	RAM	Flash
STM32F407	ARM Cortex-M4	ST Microelectronics	168 MHz	192 KB	1024 KB

Table 5.2.: *The properties of the deployed P440 ranging sensor [158]*

Accuracy	Max. operating range	Max. ranging rate	Frequency range	Transmission power
2.1 cm	300 m - 1100 m	125 Hz	3.1 GHz - 4.8 GHz	50 μW

5.1.2. Operating System

We use the RIOT-OS, which is discussed in Section 3.2.7. Based on the architecture of the RIOT-OS, we developed and integrated the device drivers for the P440 ranging sensor and the UART controller. These software driver components build the driver module and are part of the system layer (*cf.* Figure 5.2). The multithreading capability of RIOT-OS ensures that various tasks can be performed quasi-parallel. Possible tasks are distance measurement or position calculation.

5.2. Application Layer of UWB-Based Mobile Station

As illustrated in Figure 5.2, the AL is composed of two sublayers: the preprocessing and the position estimation. Both sublayers will be discussed in Sections 5.2.1 and 5.2.2, respectively. The AL incorporates a command shell for interaction with a user or interaction with an application by using the serial interface. Furthermore, the data is exchanged by using JSON format between the MS and other systems such as applications located on a PC. This can enhance the interoperability of the MS, which is achieved by using a minimalistic JSON parser at the MCU.

5.2.1. Preprocessing

As indicated in Section 4.3.3.1, the preprocessing sublayer serves to remove the outliers in the data delivered from the system layer. Therefore, we use the median filter to remove noise from the measured distances captured from the UWB sensor [160]. We apply the shell sort algorithm to implement the median filter, which does not require recursion such as the quick sort algorithm [161]. Although the iterative shell sort algorithm is slower than the quick sort algorithm, it is suitable for resource-constrained devices such as MCUs with a limited stack size.

5.2.1.1. Shell Sort

Shell sort is a sorting method, which was invented by Donald Shell and published in 1951 [162]. It is designed to remedy the deficiencies of the insertion and bubble sort methods [163]. The Shell sort algorithm divides the array of n data elements into k sub-arrays, whereby k is called the Shell size. Then it uses a sort method for the k sub-arrays called k-sort. These steps are repeated, and k is decremented until it reaches the value of 1. Given an array A of n data elements to be sorted in ascending or descending order. The Shell algorithm is described by Sengupta and Korobkin using the following steps [163]:

Step 1: Select a value for k (e.g., $k = n$).

Step 2: Divide A into k subarrays so that every subarray contains every k-th data element of A.

Step 3: Sort each of these k subarrays by using a sorting method (e.g., insertion sort).

Step 4: Decrement the value of k by using a formula or a sequence of decreasing integers.

Step 5: Repeat steps 2 through 4 until k is equal to 1 ($k = 1$).

Step 6: The data elements of A are sorted in ascending or descending order.

In the worst case, the performance of the Shell sort algorithm varies between $\mathcal{O}(n^{1.25})$ and $1.6 \cdot \mathcal{O}(n^{1.25})$. This a remarkable improvement over the insertion algorithm, which has a performance of $\mathcal{O}(n^2)$. However, with the proper choice of the Shell size k, the complexity of the algorithm decreases to $\mathcal{O}(n \cdot (log(n))^2)$ [164].

5.2.2. Positioning Algorithms

In this subsection, we initially present a multilateration approach for the position determination. Then, we describe two methods to calculate the linear least-squares problem for the ranging based positioning system. Finally, we derive the equations to compute a Non-linear Least Squares (NLS) method in a convenient form for MCUs. The NLS method enables the position optimization.

5.2.2.1. Algebraic Multilateration Method

Assume (x, y, z) and (x_i, y_i, z_i) for $i = 1, 2, ..., n$ are the coordinates of the MS and of n reference points, respectively. In addition, the measured distances between the reference points and the MS are d_i. The unknown location of the MS is the intersection of the spheres, whose equations are:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = {d_i}^2 \qquad i = 1...n \tag{5.1}$$

The system of nonlinear equations in (5.1) can be solved by different methods [165, 166, 167]. We solved it by transforming the system of equations in a matrix form [168]. The algorithm used is not related to a specific anchor, since most algorithms subtract the coordinates of a specific anchor for the linearization of the equation system. Additionally, the algorithm gives a measure of the solvability of the multilateration problem and provides a recursive least square approach to update the position [168]. The solution of the linearized system is completely determined if the distances from four reference points are known. The problem requires the estimation of the unknown position $\vec{x} = (x, y, z)$ such that:

$$\boldsymbol{A}\vec{u} = \vec{b}, \tag{5.2}$$

where $\vec{u} = (x^2 + y^2 + z^2, x, y, z)$, the matrix $\boldsymbol{A}$ and the vector $\vec{b}$ have the following forms [168]:

$$\boldsymbol{A} = \begin{pmatrix} 1 & -2x_1 & -2y_1 & -2z_1 \\ 1 & -2x_2 & -2y_2 & -2z_2 \\ 1 & -2x_3 & -2y_3 & -2z_3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & -2x_n & -2y_n & -2z_n \end{pmatrix}, \tag{5.3}$$

$$\vec{b} = \begin{pmatrix} d_1^2 - x_1^2 - y_1^2 - z_1^2 \\ d_2^2 - x_2^2 - y_2^2 - z_2^2 \\ d_3^2 - x_3^2 - y_3^2 - z_3^2 \\ \vdots \\ d_n^2 - x_n^2 - y_n^2 - z_n^2 \end{pmatrix}. \tag{5.4}$$

Equation (5.2) can be solved based on the QR decomposition in the microcontroller. An efficient method to implement the QR decomposition is the Householder transformation [169]. If $\boldsymbol{A}$ is ill-conditioned or singular, the position $\vec{x}$ can be computed using the Moore–Penrose pseudo-inverse algorithm [168, 169]. The Moore–Penrose pseudo-inverse is the best linear reconstruction operator in the mean square sense [170], which is more robust and reliable than the Householder transformation, but substantially more computationally expensive:

$$\vec{u} = \boldsymbol{A}^{+}\vec{b}, \tag{5.5}$$

whereby $\boldsymbol{A}^{+}$ is the pseudo-inverse of the matrix $\boldsymbol{A}$ [168, 169]. The pseudo-inverse matrix can be computed based on the SVD of the matrix $\boldsymbol{A}$ [171]. SVD enables the calculation of the underdetermined and overdetermined systems of linear equations. Furthermore, the SVD is more robust to numerical errors [172], but it is computationally expensive.

5.2.2.2. Preprocessed Pseudo-Inverse Matrix

Since the matrix $\boldsymbol{A}$ in (5.3) depends only on the coordinates of the RSs, the constant matrix ($\boldsymbol{A}^{+}$) can be computed externally, e.g., in a PC. This method enables saving the resources of the MS, which can be initialized with the matrix $\boldsymbol{A}^{+}$, and for example, with the help of serial communication. In this case, the computation of a new $\vec{x}$ position in Equation (5.2) is reduced to a matrix multiplication: $\boldsymbol{A}^{+}\vec{b}$.

5.2.2.3. Non-linear Least Squares Method: Gauss–Newton Method

The algebraic multilateration method does not always provide a good estimation due to the measurement uncertainties [168]. In this case, the NLS method can be used to improve the position calculated by the algebraic multilateration method. This method is based on the minimization of the squares of the errors:

$$F(x, y, z) = \sum_{i=1}^{n} f_i^2(x, y, z), \tag{5.6}$$

whereby $f_i(x, y, z)$ is the error function:

$$f_i(x, y, z) = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} - d_i \tag{5.7}$$

Minimizing the sum of the square errors is a common problem in the area of applied mathematics, which can be solved, for instance, with the Gauss–Newton or the Levenberg–Marquardt algorithms [169]. We use the Gauss–Newton method to improve the estimated position, which is calculated by using the Moore-Penrose pseudo-inverse algorithm.

Since the Gauss-Newton method requires the first derivatives, we define the following Jacobian matrix:

$$\boldsymbol{J_f} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x} & \frac{\partial f_n}{\partial y} & \frac{\partial f_n}{\partial z} \end{pmatrix} \tag{5.8}$$

We introduce the error function vector $\vec{f}$:

$$\vec{f} = \begin{pmatrix} f_1, & f_2, & f_3, & \ldots, & f_n \end{pmatrix}^T \tag{5.9}$$

Starting with an initial position guess $\vec{x}^{(1)} = (\tilde{x}, \tilde{y}, \tilde{z})^T$ calculated by the multilateration method, the Gauss-Newton method proceeds using the following iterations:

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + \vec{s}^{(k)} \tag{5.10}$$

$$\vec{s}^{(k)} = -(\boldsymbol{J_f}^{T(k)} \boldsymbol{J_f}^{(k)})^{-1} \boldsymbol{J_f}^{T(k)} \vec{f}^{(k)}, \tag{5.11}$$

where $\vec{x}^{(k)}$ is the k-th approximation of the position and $\vec{s}^{(k)}$ is the k-th error correction vector. We calculate $\vec{s}$ by using the Moore–Penrose pseudo-inverse algorithm, since the QR-decomposition, such as the QR-Householder algorithm, can fail due to the singularity or poor conditioning of the matrices.

Using equations 5.7 and 5.8 lead to:

$$\boldsymbol{J_f}^T \boldsymbol{J_f} = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x-x_i)^2}{(f_i+d_i)^2} & \sum_{i=1}^{n} \frac{(x-x_i)(y-y_i)}{(f_i+d_i)^2} & \sum_{i=1}^{n} \frac{(x-x_i)(z-z_i)}{(f_i+d_i)^2} \\ \sum_{i=1}^{n} \frac{(x-x_i)(y-y_i)}{(f_i+d_i)^2} & \sum_{i=1}^{n} \frac{(y-y_i)^2}{(f_i+d_i)^2} & \sum_{i=1}^{n} \frac{(y-y_i)(z-z_i)}{(f_i+d_i)^2} \\ \sum_{i=1}^{n} \frac{(x-x_i)(z-z_i)}{(f_i+d_i)^2} & \sum_{i=1}^{n} \frac{(y-y_i)(z-z_i)}{(f_i+d_i)^2} & \sum_{i=1}^{n} \frac{(z-z_i)^2}{(f_i+d_i)^2} \end{pmatrix}, \tag{5.12}$$

and

$$\boldsymbol{J_f}^T \vec{f} = \left(\sum_{i=1}^{n} \frac{(x-x_i)f_i}{(f_i+d_i)}, \sum_{i=1}^{n} \frac{(y-y_i)f_i}{(f_i+d_i)}, \sum_{i=1}^{n} \frac{(z-z_i)f_i}{(f_i+d_i)} \right)^T \tag{5.13}$$

Equations (5.12) and (5.13) are composed of sum terms, which can be implemented in

Table 5.3.: *Complexity of various algorithms*

Algorithm	Complexity [flops]
Matrix multiplication: $\boldsymbol{A_{m,n}} \times \boldsymbol{B_{n,p}}$	$mp(2n-1)$ [173]
QR-Householder	$2mn^2 - \frac{2}{3}n^3$ [174]
Moore–Penrose pseudo-inverse	$4m^2n + 8mn^2 + 9n^3$ [175]

the microcontroller, for example using a for-loop; whereby, the upper bound of the loop is n, which is equal to the reference point's number. The matrix in Equation (5.12) is symmetrical; this property can be used to reduce the computational burden by computing only the upper or lower part of the matrix. Finally, the terms $(fi + di)^2$ and $\frac{f_i}{f_i+d_i}$, which appear in each sum term of the matrix elements, can be computed only once by each iteration in Equations (5.12) and (5.13), respectively.

5.3. Complexity of the Algorithms Used

The algorithms used in this chapter are based on the matrix multiplication, the QR-Householder, and the Moore–Penrose pseudo-inverse algorithm. Their complexity is summarized in Table 5.3, where m and n are the number of rows and columns of the matrix $\boldsymbol{A}$, respectively. While, n and p are the number of rows and columns of the matrix $\boldsymbol{B}$, respectively.

5.4. System Evaluation

In this section, we present the results of the experimental evaluation of the UWB-based system. The aim of this evaluation is to demonstrate the feasibility to implement the proposed platform architecture for a TOA-based system, and therefore we will not address issues such as the impact of the placement of the anchors and the MS, or the selection of the anchors on the localization accuracy. The impact of the MS related to the anchors' placement and number is covered in Chapter 7. For the evaluation, we provide the results of the accuracy measurements of the UWB-based system. Furthermore, we evaluated the computing time of the algorithms on the STM32F407, which is running at 168 MHz. Finally, we evaluated the energy consumption of the algorithms running on the MCU as well as of the MS in Section 5.4.3.

A static measurement setup was performed to inspect the positioning performance of the UWB system, whereby four UWB reference transceivers were situated in the corners of a 6 m × 7 m room and the UWB mobile stations are placed in several points and at three different heights in the room. Hence, the location of the MS is measured at 27 different locations, whereby the measurement is repeated fifty times at each location. The MSs lie in

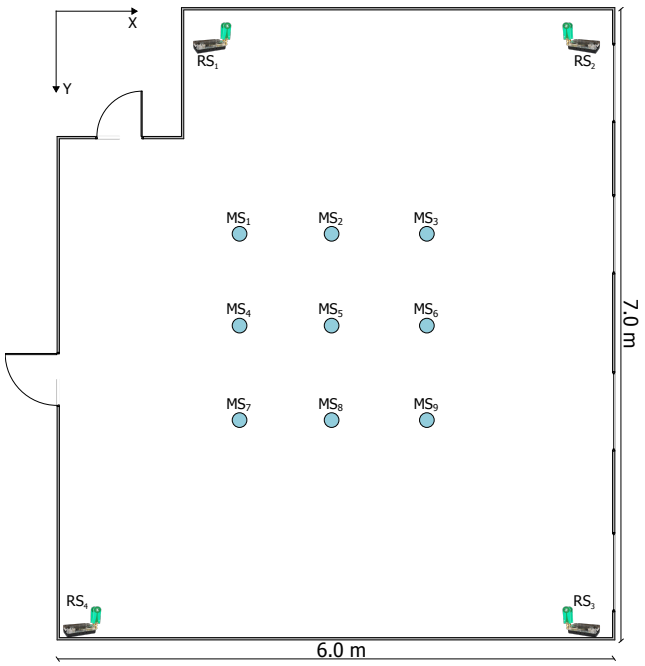


Figure 5.4.: *Experimental setup for position measures between various MSs and four reference stations*

a one-meter grid (see Figure 5.4).

5.4.1. Accuracy Evaluation

Figure 5.5(a) illustrates the three-dimensional position errors by using the algebraic multilateration and the Gauss–Newton methods. Figure 5.5(b) shows the positioning error, which is defined as the Euclidean distance between the estimated and true position. Figure 5.5(c) contains the empirical CDF of the position error of all locations by using the algebraic multilateration method; the error in the x- and y-coordinates is less than 3.5 cm, while the error in the z-coordinate is less than 25.3 cm. The error in the z-coordinate results from the unfavorable geometrical configuration, since the reference stations are located at approximately the same height. By further applying the Gauss–Newton method, the positioning error is reduced to 2.2 cm in the x- and y-coordinate, as well as to 11.2 cm in the z-coordinate (see Figure 5.5(d)).

5.4.2. Computing Time Measurement

The matrix $\boldsymbol{A}^{+}$ is calculated based on the Moore–Penrose method only once, in the initial phase at the start of the MCU or the positioning application. Based on the computed matrix $\boldsymbol{A}^{+}$, the localization of the MS is determined by the algebraic multilateration method. The position of the MS can be improved by using the Gauss–Newton algorithm, which uses the

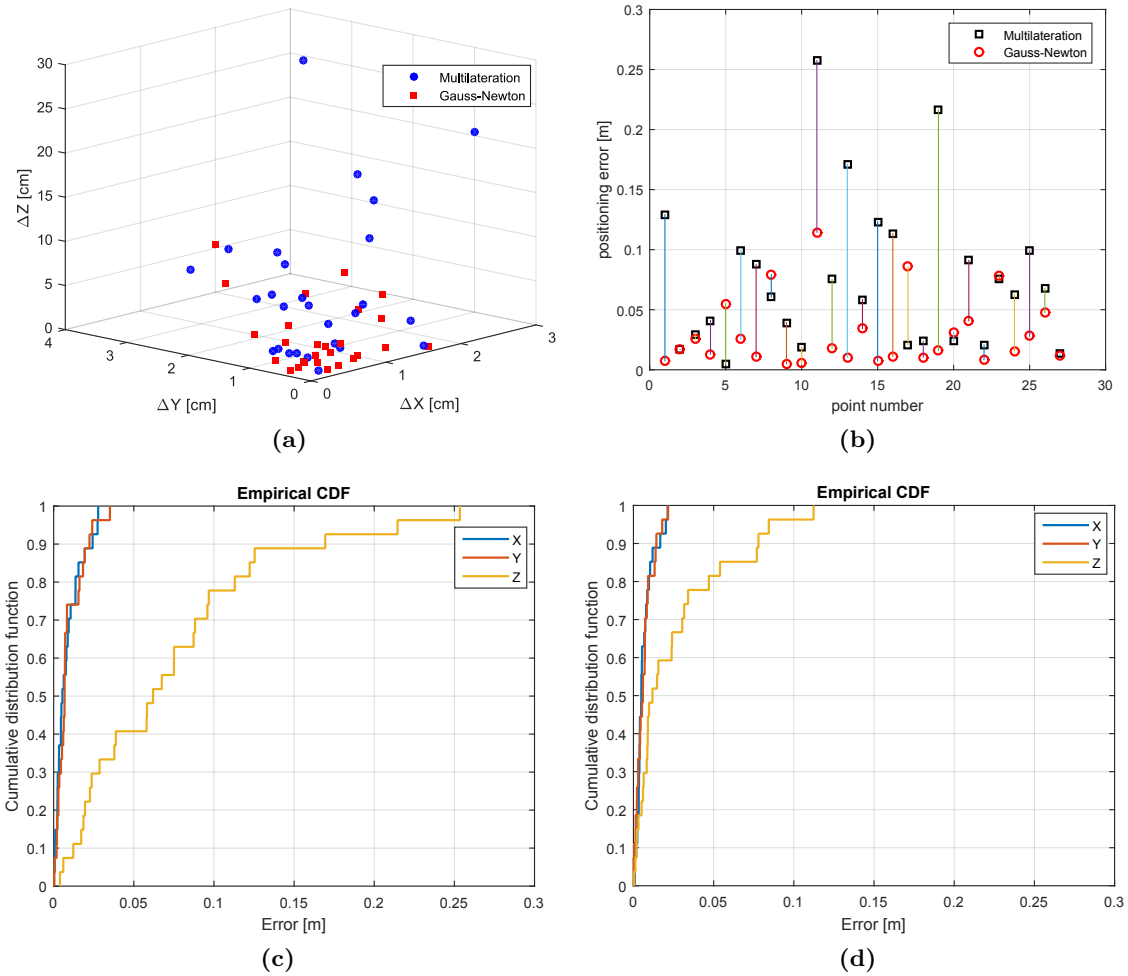


Figure 5.5.: *UWB-based system: scatter plots, position errors, and empirical CDFs. ML, Multilateration; GNM, Gauss–Newton Method. (**a**) Scatter plot of the ML algorithm and the GNM, (**b**) position error of the ML algorithm and the GNM, (**c**) empirical CDF of the points estimated by the ML algorithm, (**d**) empirical CDF of the estimated positions after the use of the GNM*

position delivered from the algebraic multilateration method as a starting point. The Gauss–Newton proceeds iteratively up to the desired accuracy or until the maximal iteration number is reached. The average iteration number in this experiment is five. Despite the constrained computing resources of the STM32F407-MCU, the computing time of each positioning is in the order of 0.032 ms without using the Gauss–Newton algorithm. In contrast, the mean estimated position time increases up to approximately 7.9 ms by using the Gauss–Newton algorithm. The evaluation of the described computing steps is summarized in Table 5.4.

Table 5.4.: *Mean computing times of the algorithms used by the UWB-based System. Computing times measured on a STM32F407-MCU running at 168 MHz*

Algorithm	Computing time [μs]
$\boldsymbol{A}^+$ for the multilateration method (at the start)	2115
Multilateration method	32
Gauss-Newton per iteration	1561

5.4.3. Energy Consumption

The energy consumption of the algorithms is measured based on the measurement of the drain-source current in the supply line, which is powered by a reference voltage supply V_{cc} ($V_{cc} = 5\,\mathrm{V}$). Hence, the energy used for each localization processing task can be calculated by integrating the electric power over the times, which are summarized in Tables 5.4. We measured a current consumption of about 75 mA at the ambient temperature of 26 $^\circ C$ for the STM32F407-MCU in an active mode. The measured energies for the localization algorithms by the UWB-based ILS are summarized in Table 5.5.

Table 5.5.: *List of measured energy consumption values of the algorithms by the UWB-based ILS*

Algorithm	Energy [μWs]
$\boldsymbol{A}^+$ by the multilateration method (at the start)	793.13
Multilateration method	12
Gauss–Newton per iteration	585.38

We also measured the energy consumption of the MS performed for the UWB-based ILS by using the previously mentioned method for the energy consumption of the algorithms. We measured a current consumption of about 410 mA by the UWB transceiver; whereas, the measurement time by the UWB-based ILS is 120 ms. The total energy, required for a position estimation, is calculated based on the current drain of the sensors, as well as the energy consumption of the MCU (see Table 5.5). The energy usage of the UWB-based ILS is summarized in Table 5.6, whereby the energy consumption of the UWB transceiver is 246 mW.

Table 5.6.: *List of measured energy consumption of the UWB-based ILS for a position estimation*

Localization System	Energy [mWs]
UWB-based ILS	$246 + 0.793 + 0.012 + 5 \times 0.585 \simeq 249.73$

5.5. Conclusion

The proposed platform enables position finding based on UWB technology as well as the use of a low-cost device to implement decentralized computing. Compared with a smartphone, the localization of an object or a person is independent of its orientation and position. Furthermore, to provide for an accurate localization, the platform enables both smooth localization and the deployment of various, accurate measurement devices. The use of the RIOT-OS and standard formats extends the interoperability of the system with other protocols and systems such as the IoT. Finally, the energy evaluation shows that the UWB transceiver, which is the High Frequency (HF) part of the MS board, consumes the greatest amount of energy.

The use of the Gauss–Newton algorithm improves the altitude (z-component) computed by the algebraic multilateration method. The altitude errors are the result of a bad or faulty configuration of the MS to the anchors: very small or huge angles between the signals reaching the MS.

CHAPTER 6

Decentralized Magnetic Indoor Positioning System

Decentralized magnetic indoor localization is a sophisticated method for processing sampled magnetic data directly on a mobile station and thereby decreasing or even avoiding the need for communication with the base station. In contrast to central-oriented positioning systems, which transmit raw data to a base station, decentralized indoor localization pushes application-level knowledge into the MS. A decentralized position solution therefore has a strong feasibility to increase energy efficiency and to prolong the lifetime of the MS.

Numerous technologies for indoor positioning have been developed over the years. In [49, 10, 22, 32], a comparison of different technologies is provided in regard to accuracy, coverage, update rate, hardware size and cost. The main challenge of these technologies is the signal shadowing due to the presence of obstacles between the transmitter and receiver. Unlike other technologies, magnetic signals can pass through obstacles without significant propagation errors, even in NLoS scenarios. However, magnetic signals show a limited coverage area, since the magnetic field strengths decay rapidly with distance. Hence, large coils and high-power levels are required to reach a wide coverage.

To design a robust positioning system in challenging indoor environments, it is of paramount importance to push the application-level data processing as deeply into the MS as possible and to use a localization technology, which overcomes the limitations of existing indoor positioning systems. The processing and the evaluation of sampled data close to the source reduce the communication with the base station and minimize the energy consumption of the mobile station. The decentralized magnetic positioning system follows this strategy by designing the mobile station in such a way that the magnetic field data are at first gathered, preprocessed, synchronized, and finally computed on-the-fly to provide the spatial coordinates of the MS.

In this chapter, we present a complete architecture and the implementation of a decentralized magnetic positioning system [13]. Furthermore, we introduce a technique for the synchronization of the observed magnetic field on the MS with the artificially-generated magnetic field from the coils. Based on Real-Time Clocks (RTCs) and a preemptive operating system, this method allows a stand-alone control of the coils and a proper assignment of the magnetic fields measured on the MS. A stand-alone control and synchronization of the coils

and the MS have an exceptional potential to implement a positioning system without the need for wired or wireless communication and enable a deployment of applications for rescue scenarios, like localization of miners or firefighters. Furthermore, we discuss the localization algorithms related to the decentralized MILPS.

MILPS is based on DC-pulsed magnetic signals that show no special multipath effects and have excellent characteristics for penetrating various obstacles [176]. Therefore, MILPS offers various benefits in comparison to other active positioning systems. In this work, we propose a stand-alone localization system that enables a positioning in harsh conditions without the need for communication infrastructure, nor fixed or tedious installations. The main contribution of this work is the proposal of a decentralized control of the individual coils (anchors), as well as the decentralized synchronization of the entire system without the need for communication technology. Both the synchronization and the control of the coils and MS are based on a preemptive real-time operating system and RTCs. The algorithms for the localization as well as for the optimization are adapted for resource-constrained MSs. Furthermore, the developed work can be summarized as follows:

1. The design of a decentralized positioning system by improving the MILPS and using Coil Driver Units (CDUs), which are based on accurate RTCs. Furthermore, the MS is extended with a sensor platform including a magnetic field sensor and an RTC. The MS operates independently from the CDUs, and no communication channel is required.

2. The application of TDMA for the generation of periodic, distortion-free magnetic field signals for a certain time period (e.g., 1 s). The TDMA allows the MS to distinguish between the coils (reference points).

3. The evaluation of two approaches to drive and synchronize the coils.

4. The presentation, analysis, and evaluation of the algorithms for MILPS.

The remainder of the chapter is organized as follows: firstly, we review related works, then we present MILPS as proof of concept in Section 6.1. We introduce a new, decentralized version of MILPS in Sections 6.2 and 6.3, as well as a decentralized synchronization approach, which is based on precise RTCs in Section 6.4. Then, we describe the architecture of the CDU in Section 6.5. In Section 6.6, we discuss the algorithmic core of the system. We give an experimental evaluation of the system in Sections 6.7 and 6.8. Finally, we conclude this chapter in Section 6.9.

6.1. Related Work and Proof of Concept

This section is devoted to an overview of related work on positioning based on artificially-generated magnetic fields, and to a review of our previous work as proof of concept.

The emphasis is on summarizing the state of the art by focusing on the need for time synchronization and the architecture.

6.1.1. Related Work

As mentioned in Chapter 2, the localization systems use various physical principles and exhibit different performance characteristics. However, contrary to electromagnetic waves, magnetic signals can pass through almost all building materials without significant attenuation or distortion and are generally convenient for indoor localization purposes. Magnetic indoor positioning systems are classified into three categories: fingerprinting (geomagnetic), permanent magnet- and current-based magnetic positioning systems.

Fingerprinting positioning systems such as presented in [74, 177] require no synchronization as well as no infrastructure such as coils. Only a three-axis magnetometer is used to achieve a self-localization, which can be decentralized computed, for example, on a smartphone. The position can be also centralized computed on a server such as the case in [177]. These systems are restricted to one-dimensional locations (e.g., location of a person within a corridor). The acquisition of magnetic maps in the setup phase is indispensable, whereby the localization accuracy scales with the finger printing data resolution.

Permanent magnets can be used to implement a magnetic positioning system as presented by Song *et al.* [75]. Song *et al.* use a permanent magnet enclosed in a capsule as a MS and a magnetometer array as reference points. Most of these systems use a centralized unit such as a PC or a microprocessor board to synchronize the measured magnetic data as well as to compute the position [75, 76]. This class of magnetic positioning system has a restricted coverage volume (up to $1\,\mathrm{m}^3$) and uses complicated mathematical models with high-order non-linear equations [49].

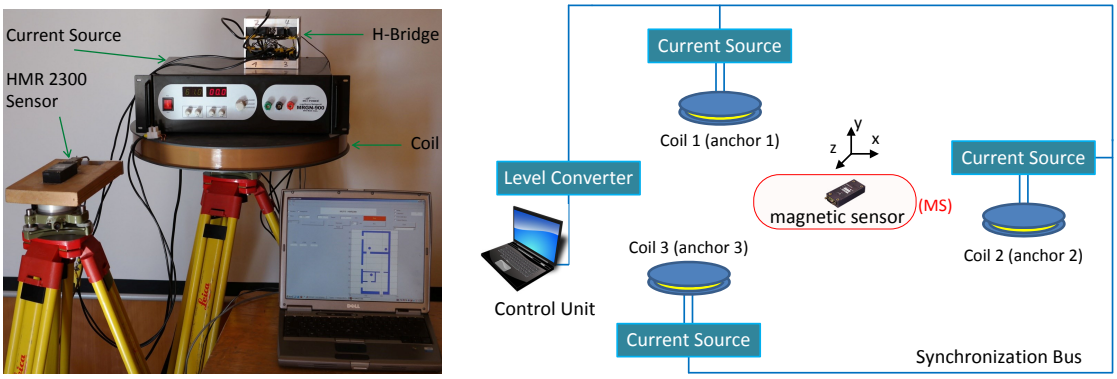
The third class artificially generates magnetic signals by means of coils with known positions (reference points), which generate magnetic fields by using pulsed DC or AC. The current-based magnetic positioning systems are deployed in industrial and bio-medical applications and use a centralized approach to control the coils, synchronize the collected magnetic field data, as well as to perform the localization estimation. In other words, the central unit is responsible for the synchronization between all connected components [81, 82]. Sheinker *et al.* propose a 3D positioning system, which generates a time-varying magnetic field by exciting coils with an AC source. The MS can distinguish between the beacons by using a lock-in amplifier, since each beacon is assigned a specific frequency. This method is similar to the Frequency Division Multiple Access (FDMA) approach. The positions are calculated on a centralized PC on top of the measured magnetic field amplitudes and phases [83].

Similar to our previous proof-of-concept, two experimental systems are introduced in [78, 178], which utilize coils placed at different positions to reach a wide coverage area. Since

these systems are still in the prototype phase, a centralized approach is also used. Prigge presents a prototype system that utilizes several coils [78]. A Code Division Multiple Access (CDMA) approach is used to distinguish each generated signal. A timing box generates the synchronization signal, which is distributed over a cable network to all coils and via either a wireless or a wired connection to the MS. A synchronization method is proposed in [178], which uses edge detection within the captured magnetic field signal to correct the time drift between the coils and the MS. However, this concept faces many difficulties if the captured signal is weak or the time drift is bigger than a certain amount of time, which leads to a degradation of the location accuracy. Moreover, as in other cross-correlation-based approaches, because of the uniformity of the coils switching pattern, the MS is not able to distinguish every single coil [178].

6.1.2. Proof of Concept

The objective of our proposed MILPS is to provide a reliable and accurate indoor positioning system that covers an entire building with a minimum of infrastructure and complexity. The system consists of several coils placed inside or outside the building and a mobile sensor (*cf.* Figure 6.1).

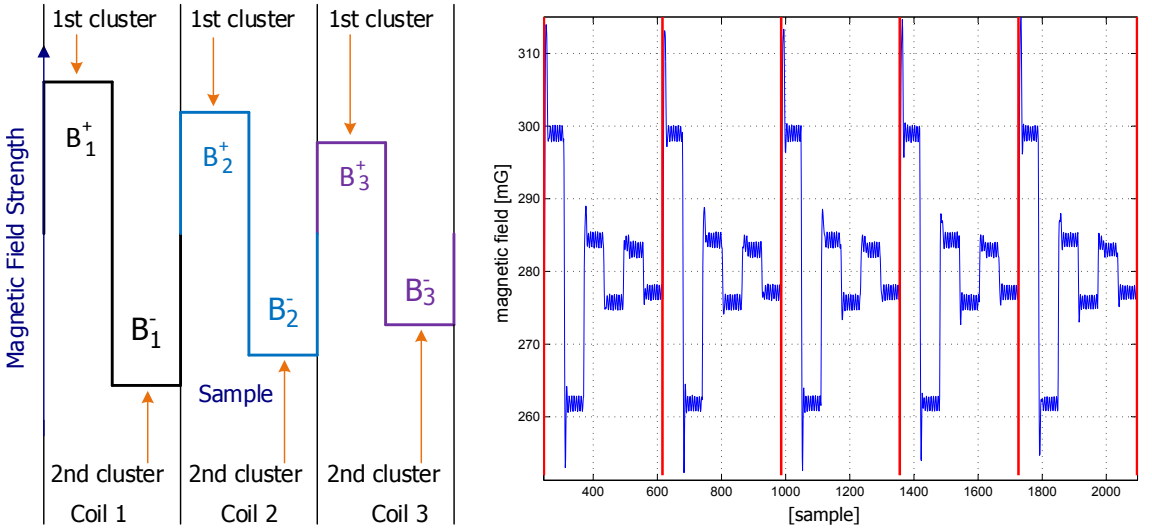


(a) **(b)**

Figure 6.1.: *Magnetic Indoor Local Positioning System (MILPS) platform. (**a**) Main components; (**b**) Basic system overview with three coils and a mobile sensor (MS)*

The coils generate magnetic fields successively. By measuring the field components of multiple coils (at least three) and using the coil coordinates in the building reference system, the unknown 3D coordinates of the MS can be estimated by applying the trilateration principle [168]. A simple theoretical and real example of a received magnetic field at the MS is presented in Figure 6.2.

As shown in Figure 6.2(a), the direction of the electrical current of each coil is switched in polarity to eliminate the overlying magnetic field of the Earth and other long periodic



(a) *Theoretical magnetic field* **(b)** *Real captured magnetic field of Coils* 1, 2 *and* 3

Figure 6.2.: *Captured magnetic fields*

magnetic interferences. This is achieved by computing the differences between subsequent positive and negative sample clusters ($B_i^+ - B_i^-$), where B_i^+ and B_i^- are the calculated medians of the first and second cluster, respectively. In addition, the magnetic field (B_i) measured from the i-th coil is computed as follows:

$$B_i = \frac{B_i^+ - B_i^-}{2}. \tag{6.1}$$

A proof of concept was introduced in [179] by presenting a working prototype. The results of the measurements performed with the prototype prove the feasibility to determine the 3D position of a user or an object inside a building, even in NLoS conditions. The prototype is based on artificially-generated magnetic fields and achieves a positioning accuracy of less than 0.5 m. The coils maintain synchronization through a communication link, which is implemented as a cable or a wireless link. The MS is also synchronized with the coils and can therefore distinguish the coil fields. The synchronization acquisition at the MS is computed by using the cross-correlation between the receiving signal and a template square signal. This method is, however, limited to stationary devices, because the square wave pattern at the MS is distorted during movement. A so-called "stop and go" measurement must be performed. That means, to regain the synchronization, the mobile station must regularly stop. This is unpractical in a real tracking scenario. To deal with this limitation, a complete centralized solution has been applied by sending the sensor raw data to a base station, which activates the coils and performs the position calculation. However, this method would face difficulties in a real indoor scenario, such as power consumption, low reliability of the data transfer, data congestion, when various MSs are involved, *etc.*

Although localization systems based on magnetic fields may be valuable for cluttered environments, they are often limited by strict synchronization requirements. Accordingly, there is a need for an improved synchronization system that has low power requirements, requires no special synchronization hardware and is easy to implement in a variety of scenarios. Thus, it is necessary to utilize synchronized clocks for both the transmitting coils and the receiving MSs.

6.2. Architectural Overview

The decentralized MILPS is based on CDUs for a stand-alone control of the coils and a MS for on-the-fly computing of the position. However, the resource constraints of the MS pose various challenges that should be considered during the design phase. Typical resource constraints are: restricted processing power, limited storage capacity, and limited energy resources. The design constraints depend on the application and the environment in which the MILPS is deployed. An important design factor that should be carefully treated is the energy consumption of a MS, since MSs are mostly battery operated. The previous considerations, like the resource constraints, the energy consumption of the MS and the system topology, provide additional inputs for the system design and architecture, as detailed in Sections 6.3 through 6.6.

Expanding on the principles of MILPS and the architectural view presented in the previous Chapter 4, the focus is shifted from the system topology aspects to the MS and the CDUs' design aspects. Based on the general discussion of architectures, MILPS principles and the synchronization problematic of the system, we now present an exemplary platform for a decentralized and synchronized magnetic positioning system. This platform is employed in MILPS, whereas the mobile station and the anchors are equipped with real-time clocks, and the MS additionally incorporates a magnetic sensor. The MS as well as the CDU are designed in a layer-based architecture.

MILPS is a decentralized and magnetic-based positioning system, which is representative for an RSS-based localization system. The MILPS developed in this work enables the calculation of an optimized three-dimensional position on the MS based on the measurement of the magnetic field as well as the elevation angle to the coils as anchors (see Figure 6.3).

As illustrated in Figure 6.4, the system architecture is divided into two layers: the application and the system layer. The AL is the highest level of the MS or CDU, which follows a modular-based architecture that ensures the portability and the extensibility of the system. In the case of the MS, it is subdivided into two sublayers: the preprocessing and the position computing layers (*cf.* Figure 6.4(a)). The first sublayer includes the data filtering module, which removes statistical outliers from the data delivered from the system layer. The second preprocessing module provides a calibration routine to correct inaccuracies of the magnetic samples. The calibration routine is not addressed in this work. The top-level

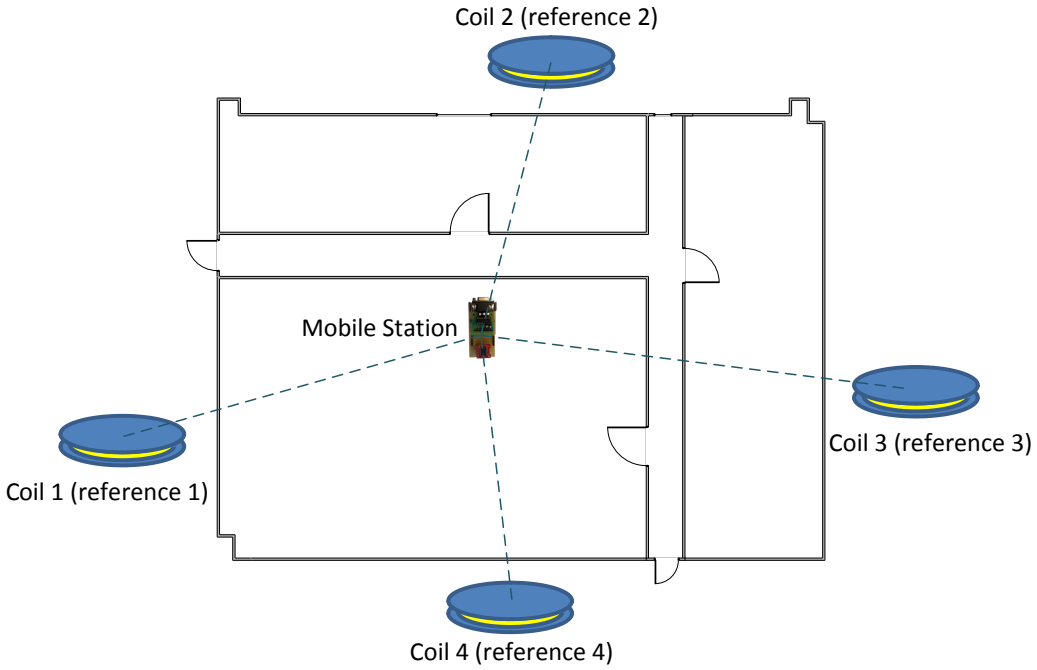


Figure 6.3.: *MILPS principle*

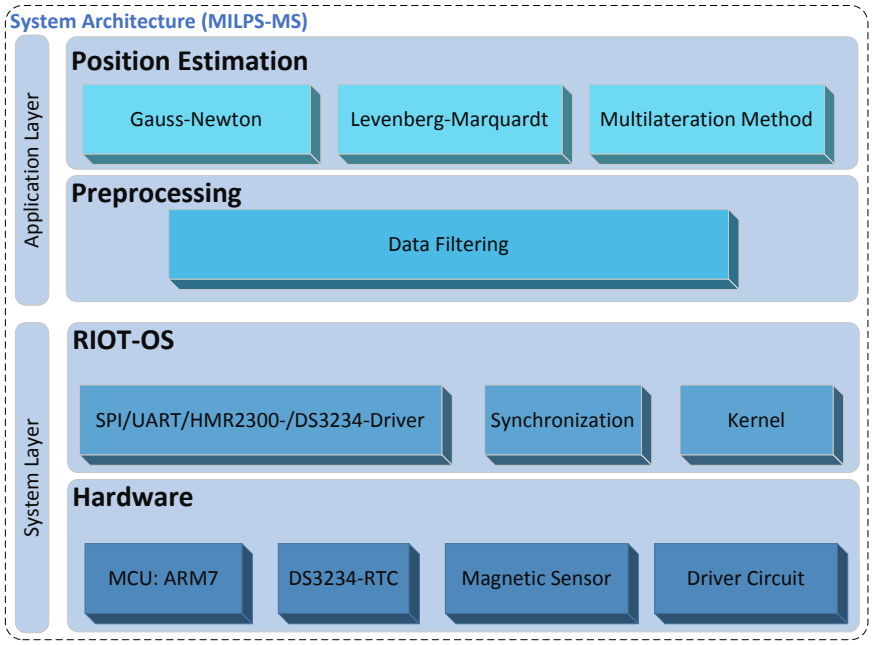
sublayer represents the algorithmic core that computes the position on the MS. In the case of the CDU, the AL includes an application for controlling the coils (*cf.* Figure 6.4(b)). Both ALs incorporate a command shell for the interaction with a user or an application by using the serial interface.

6.2.1. Layers Interaction of the Mobile Station

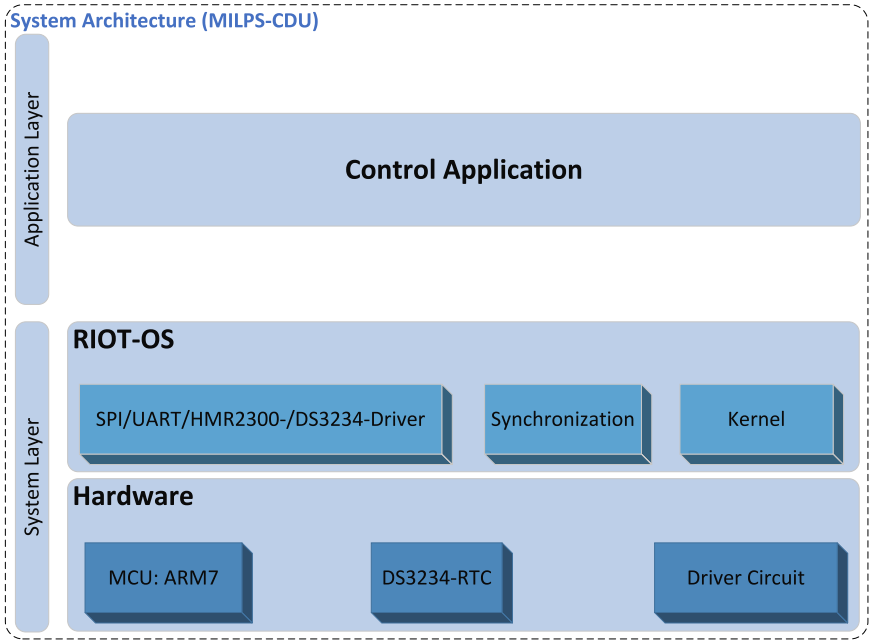
The magnetic field data are gathered as well as synchronized by the SL after performing measurements of the magnetic field strengths to the anchors. These measurements as well as their synchronization are supported by the RIOT-OS that controls the hardware sublayer including the magnetometer. The collected measurement data are delivered to the preprocessing sublayer for removing outliers and calibrating magnetic data. Finally, the position is calculated by the position-estimation sublayer based on the data delivered from the preprocessing sublayer (see Figure 6.4(a)).

6.2.2. Layers Interaction of the Control Driver Unit

The coils are controlled via a driver circuit that is connected to an H-Bridge. The control and the synchronization of the coils are performed by software drivers based on the RIOT-OS. The control application layer enables the initialization, configuration, and control of the CDUs by using the underlying SL (see Figure 6.4(b)).



(a)



(b)

Figure 6.4.: *System architecture of (**a**) the Mobile Station (MS) and (**b**) the Coil Driver Unit (CDU)*

6.3. System Layer of MILPS Mobile Station

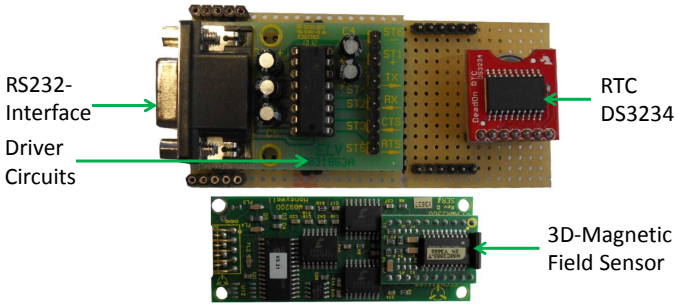
The SL is based on the previously discussed architecture template (see Figure 4.3). The SL includes the hardware board, the power unit, and the RIOT-OS [12]. The RIOT-

OS is a tickless real-time OS that supports multithreading and priority-based preemptive multitasking (*cf.* Section 3.2.7).

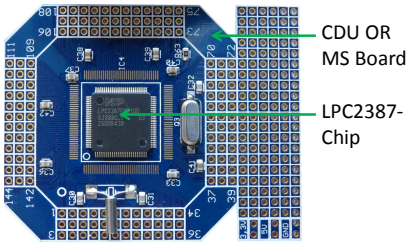
6.3.1. Hardware

The hardware is composed of four subsystems: the *power unit* supplying the sensor board with the energy, the *MCU*, the *sensing unit*, and the *driver circuits*. The MCU forms the core of the system controlling the other three subsystems. The MCU is based on an ARM7 core, operating at 72 MHz and has a memory capacity of 96 KB RAM and 512 KB Read-Only Memory (ROM) (*cf.* Figure 6.5(b)).

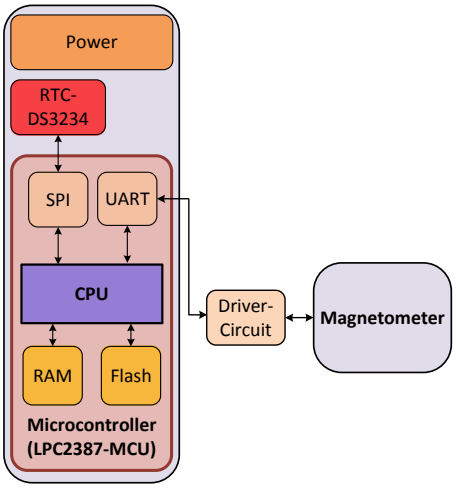
The sensing unit includes the 3D-magnetic sensor HMR2300, which offers a sampling rate of up to 154 Hz and a range of ±2 gauss (G), with a resolution up to 70 µG [180]. The sensing unit can be extended with additional sensors, like a 3D-accelerometer or gyroscope sensors to support a navigation scenario in indoor environments. The hardware of the MS is illustrated in Figure 6.5. The properties of the LPC2387-MCU as well as of the HMR2300 magnetometer used are summarized in Table 6.1 and 6.2, respectively.



(a) *Mobile Station (MS) hardware.*



(b) *MCU board for MS or CDUs.*



(c) *Simplified hardware block diagram.*

Figure 6.5.: *MILPS: MS hardware overview. MCU, Microcontroller Unit. CDU, Control Driver Unit.*

Table 6.1.: *The properties of the deployed LPC2387-MCU. MCU, Microcontroller Unit [181]*

MCU	Family	Vendor	Frequency	RAM	Flash
LPC2387	ARM7	NXP	72 MHz	96 KB	512 KB

Table 6.2.: *The properties of the HMR2300 magnetometer. FS, Full Scale [180]*

Range	Sample Rate	Resolution	Accuracy
±2 gauss (G)	up to 154 Hz	up to 70 μG	0.5% FS (over ±1G)

6.3.2. Operating System

Based on the architecture of the RIOT-OS, we develop and integrate device drivers for the DS3234 real-time clock, the HMR2300 magnetometer, the SPI bus and the UART controller. These software driver components build the driver module and are part of the system layer (*cf.* Figure 6.4(a)). Due to the support of fast interrupt handling (low interrupt latency) and multi-threading, the RTC and HMR2300 drivers run in concurrent threads, thus, enabling the synchronization of the sampled magnetic data on the MS. Because of the energy efficiency, RIOT-OS enables the development and deployment of energy-aware applications on resource-constrained devices. Further energy savings can be achieved by using MCU-specific power management techniques, which improve the lifetime of the MS.

The modular structure of the RIOT-OS allows not only the development of a magnetic-based positioning system, such as MILPS, but also the deployment of localization systems that use other technologies. The modularity and the developer-friendly features encourage the reusability of common modules, e.g., filtering algorithms. The reusability of source code is improved by defining clear interfaces between the individual layers and sublayers. Therefore, MILPS can be easily extended with other technologies. By fusing data from different types of sensors, it is possible to compensate for the shortcomings of a single technology, such as coverage gaps, or to improve the positioning performance.

6.4. Synchronization

Synchronization plays a key role for MILPS, since the coils are activated in successive time slots (TDMA), and the MS has to classify the sampled magnetic data into the proper coil at the corresponding time. In general, a vital element for proper operation is a reliable clock source, which is essential for microcontrollers and RTCs. Since the oscillators are the basis for the MILPS synchronization sources-clocks, they will be discussed in combination with timing and synchronization on resource-constrained devices in the next Sections 6.4.1

and 6.4.2, respectively. Furthermore, we introduce a decentralized synchronization approach based on a high precision RTC, as well as two synchronization methods of the CDUs in Sections 6.4.3 and 6.5.2, respectively.

6.4.1. Stability and Accuracy of Oscillators

Oscillator circuits generate a sinusoidal or a square output signal. They are classified in RC-, LC-, or quartz crystal oscillators. The quartz-based oscillators are preferred, since they are more precise and accurate over a wider timeframe and temperature range compared to other oscillator classes. Furthermore, they have a high Quality Factor (QF), which is equal to the resonance frequency divided by the resonance width [182]. The QF of quartz-based oscillators is in the range of 10^2 to 10^6. In contrast, a LC oscillator has a maximum QF of 10^2. The quartz crystal is the core component of a crystal oscillator, which exhibits mechanical and electrical properties. In other words, mechanical forces applied to the quartz will generate an electrical field, conversely, electrical charges produce mechanical forces. This phenomenon is known as the piezoelectric effect.

The quartz crystal is affected by factors such as the temperature, humidity, power supply voltage and mechanical shocks [183, 184]. Furthermore, the frequency and stability can be negatively impacted by the following factors:

- *Aging*: is a gradual change in frequency over a long period of time due to electromechanical effects such as mass transfer and stress in the quartz.
- *Short-term instabilities*: are standard deviations of the fractional frequency fluctuations for a specific averaging time that are random in nature and are referred to as noise.
- *Phase noise*: is the random fluctuations in the phase component of the output signal [185].
- *Temperature drift*: means that the resonant frequency of the crystal varies depending on the ambient temperature; consequently, high temperatures affect the nominal frequency, which can reach a deviation of up to a few tenths of parts per million (ppm) of seconds [185].

Based on the quartz crystal and electronic components such as capacitor, resistor and transistor, various oscillator circuits can be configured. The common crystal-based oscillator circuits are Pierce-, Colpitts-, and Hartley-Oscillator [186].

6.4.2. Timing and Synchronization on Resource-Constrained Devices

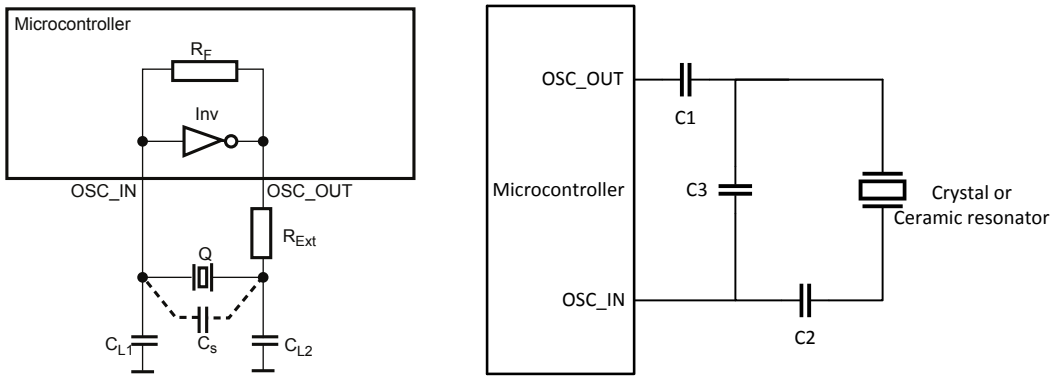
The passing time is crucial for a resource-constrained device such as a microcontroller, since many tasks are time-driven or periodic such as data reading or balancing a scheduler queue.

A full-fledged OS is a general-purpose OS that guarantees the best resource allocation for all running processes by a *Completely Fair Scheduler (CFS)*, which is not adapted to the requirements of real-time systems [187]. Although full-fledged devices are not resource-constrained, they are unsuitable to perform real-time periodic tasks such as controlling the coils periodically, since full-fledged OSs are usually not preemptive [187, 188]. The real-time characteristics of a full-fledged OS (e.g., Linux) can be improved by relatively complex extensions [187, 188]. In the following paragraphs, we discuss the realization of the timing and synchronization on microcontrollers.

The time can be specified as a *relative* and *absolute* time. Relative time is usually needed to schedule an event for one second in the future, conversely managing the current time of day requires a concept of absolute time. There are two types of clocks, the *hardware* and the *system clock*. The first clock runs independently of the MCU and even if the device is powered off. This clock provides a non-volatile unit for storing the wall-time, which is called the RTC clock and is built into most modern MCUs [189, 190]. The second clock is based on the system timer (MCU timer) which raises an interrupt at a preprogrammed frequency called the *tick rate*. The *tick* is the time between two successive timer interrupts and is equal to $\frac{1}{tick\,rate}$. A global variable holds the number of ticks since the system was booted. This variable is initialized from the kernel when the system boots up and is incremented by one during each timer interrupt. On the other hand, the kernel uses the RTC to initialize a second variable at start-up. By updating both variables inside the timer interrupt handler, the kernel can keep track of both wall and system time.

Modern microcontrollers incorporate an internal RC oscillator which is often inaccurate and sensitive to the supply voltage and temperature variations. Therefore, an external crystal oscillator is used for improved stability, frequency accuracy, low power consumption, and flexibility of a wide choice of frequency values [191]. The most common oscillator configurations for microcontrollers are the Pierce and the Colpitts circuits [192, 193] (see Figure 6.6). These circuits build external clock sources for the components of the microcontroller such as hardware timers. Timer architecture for driver and application programming are based on these hardware timers.

Constrained OSs support timers with high resolution (up to $1\,\mu s$), which are more convenient for timing measurement or performing a task during a given time interval. Nonetheless, they do not allow the measurement of absolute time. The absolute time plays a key role in the synchronization of MILPS, since we must implement periodic tasks that rely on the concept of absolute time. A further requirement is the kernel preemption that enables application processes to preempt the kernel. Conversely, the synchronized tasks can be strongly affected by other activities such as a disk or network traffic in a system, which does not support preemption, especially when it is exposed to a heavy load. Nevertheless, due to the temperature drift or aging, the source clock remains the weakest part of the system (*cf.* Section 6.4.1).



(a) *Pierce Oscillator [194].* **(b)** *Colpitts Oscillator [193].*

Figure 6.6.: *Commonly used oscillators for microcontrollers*

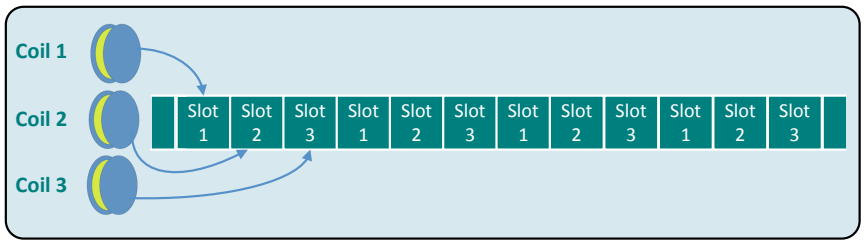


Figure 6.7.: *MILPS: system synchronization (TDMA)*

6.4.3. Synchronization Scheme Based on Real-Time Clocks

Based on the general discussion of the oscillators' stability and synchronization on resource-constrained devices, we now present a decentralized synchronization mechanism, which implements the periodic control of the coils and the MS using the TDMA scheme (*cf.* Figure 6.7). The MS is synchronized with the CDUs without the need for a synchronization bus or any kind of communication.

The goal is to generate time slots with a certain period (e.g., 1 s) or multiples thereof, with a tolerance range of ±20 %, which must not be exceeded during a total operating time of 2–3 h (e.g., the period of a fire rescue operation). The core of our method is based on the DS3234 low-power and accurate RTC with the accuracy of ±2 ppm or ±3 ppm for the temperature ranges from 0 °C to 40 °C or from −40 °C to 80 °C, respectively. The performance of the DS3234 is achieved by the combination of a Temperature Compensated Crystal Oscillator (TCXO), which provides a high level of temperature stability, and the RTC [195].

The built-in TCXO offers an accurate and stable reference clock, which keeps the accuracy of the RTC within ±2 min per year in a temperature range from −40 °C to 80 °C [195]. In addition, the TCXO provides a square-wave with four programmable frequencies from 1 Hz

to 8.2 kHz and a battery backup unit to continuously keep the time and settings in 256 bytes of Static Random-Access Memory (SRAM).

For a typical deployment, MILPS operates in three distinct phases:

1. The initialization phase: Both the RTCs of the CDUs and the MS are set to the same time. This phase is initiated by a user with the help of a serial splitter or wireless connection. The RTC is integrated into the CDU and the MS via the SPI-interface. Once the initialization is complete, the system enters the operating mode.

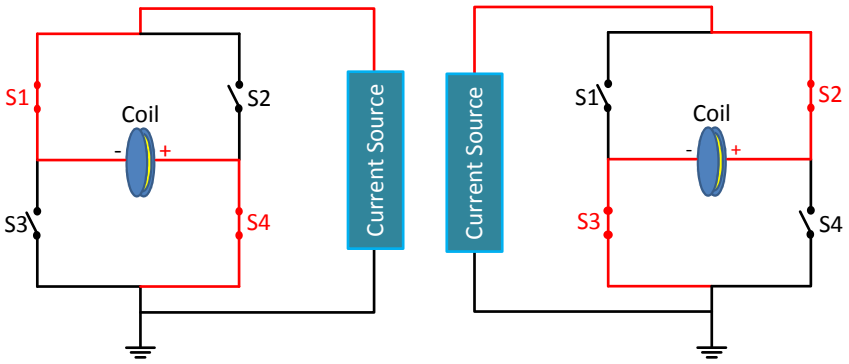
2. The operating mode: In this mode, each coil's CDU is assigned to a fixed duration slot, in which the coils are activated. Like TDMA, the time slots are cyclically organized (*cf.* Figure 6.7). Simultaneously, the MS acquires the magnetic data from the HMR2300 sensor, which can be assigned to the source coils by means of the RTC and predefined time windows. Based on the RIOT-OS, which provides a preemptive kernel scheduler and a fast interrupt handling, the data sampling and the time division run in different threads. Despite the clock drifts of the temperature-compensated RTCs in this phase, the distance measurement is only affected if a certain timeframe elapses. The maximum time threshold, which affects the distance measurement, is examined in the experimental part and is in the order of 80 h (see Section 6.7).

3. The resynchronization phase: The clocks of the MS and the CDUs are reset to the same time before the maximum time threshold elapsed.

6.5. Control Driver Unit

The CDU also follows a layered and a modular-based architecture (see Figure 6.4(b)). The driver circuits integrated into the hardware sublayer enable the CDU boards to drive the coils, which generate a square-wave signal as illustrated in Figure 6.2. The coils are controlled via a CDU through a Solid State Relay (SSR)-Unit, which is interfaced with a driver circuit. The SSR-Unit consists of four relays that have a maximum turn-off/on time of 300 μs/1.0 μs. The four SSR switching elements form an H-Bridge to control the voltage polarity and to enable a galvanic isolation between the control and the high voltage load circuit without the use of mechanical parts. Commonly, the SSRs can achieve a fast switching frequency up to 10^8 Hz and a response time up to 1 ns, and have no limited lifetime (number of operations) [196]. Therefore, the SSRs are more advantageous compared to the electromechanical relays. A simple schematic in Figure 6.8 showing four switches connected in a bridge configuration, whereby the switches 1 and 4 are closed and the switches 2 and 3 are kept open, causing a current flow from the power source through switch 1, the coil, and switch 4. This leads to the generation of a clockwise magnetic field. Analogously, the opposite magnetic field is

induced by closing the switches 2 and 3. Both the MSs and the CDUs are equipped with RTCs to realize a distributed synchronization as described in the previous Section 6.4.3.

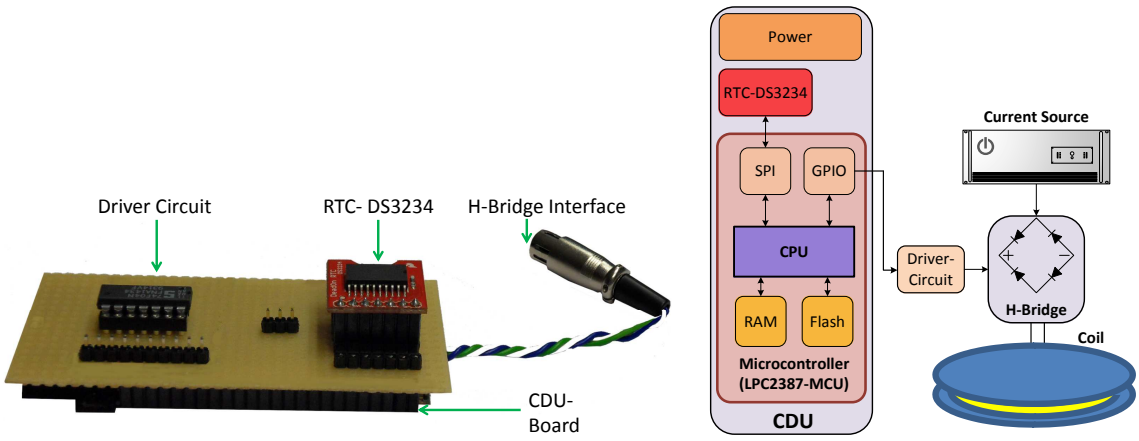


(a) *H-bridge reverse* **(b)** *H-bridge forward.*

Figure 6.8.: *Coil control for generating a bipolar magnetic field over a H-Bridge*

6.5.1. Hardware

Each coil is driven via a CDU, which includes a LPC2387-MCU, an RTC and a driver circuit. The driver circuit enables the CDU to interface with the H-bridge to control the voltage polarity. Figure 6.9 illustrates the CDU, as well as the connection of the CDU with the coil and the current source.



(a) **(b)**

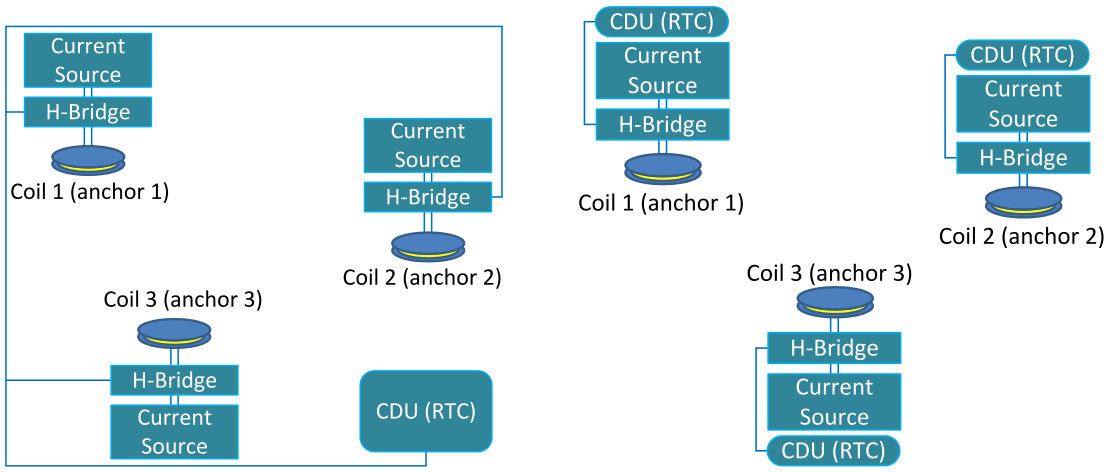
Figure 6.9.: *MILPS reference stations hardware. (**a**) Control Driver Unit (CDU), (**b**) simplified hardware block diagram*

6.5.2. Coil Synchronization Schemes

Independent of the synchronization of the MS, the coil synchronization is based on RTCs and can be used in two configurations:

1. The One-CDU configuration in which the coils are driven by the same CDU. This configuration is less sensitive to the clock drifts, because the drift occurs only between two clocks: the CDU and MS clocks (*cf.* Figure 6.10(a)).

2. The N-CDU configuration in which each coil is driven by a separate CDU, whereby N is the number of coils. This synchronization is more sensitive to the clock drifts than the One-CDU configuration, since the time drift can occur between each CDU- and MS-RTC and between the CDU-RTCs in the neighboring time slots (*cf.* Figure 6.10(b)).



(a) *One-CDU configuration.* **(b)** *Three-CDU configuration.*

Figure 6.10.: *Synchronization configurations for three coils*

6.5.3. Application Layer

The AL of the CDUs includes an application to drive the coils in the One- or the N-CDU configuration (see Figure 6.4(b)). This application enables the assignment of the time slots, the adjustment of the activation time, and the control of the coils. The AL allows for the initialization of the RTCs by using wired or wireless communication. The AL incorporates a command shell to interact with the user or an application. Furthermore, the state of each CDU such as the time or temperature can be requested by using the serial or wireless communication.

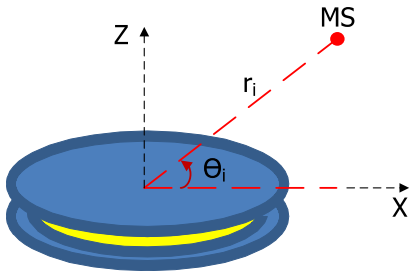


Figure 6.11.: *Elevation angle Θ_i of the MS relative to the coil plane*

6.6. Application Layer of MILPS Mobile Station

The AL is the highest level of the MS, which follows a modular-based architecture. It is subdivided into two sublayers, the preprocessing and the position computing sublayers (*cf.* Figure 6.4(a)). The first sublayer includes the data filtering module, which uses the median filter to remove the outliers from the gathered magnetic data delivered from the system layer. The top-level sublayer represents the algorithmic core that computes the position on the MS. The AL also incorporates a command shell for the interaction with a user or an application by using the serial or wireless interface like the UWB-based system. Furthermore, the AL of the MS includes a minimal JSON parser. In the following section, we describe the algorithms for the position estimation.

Theoretically, the magnetic field B_i generated from the coil i, is given by the following equation:

$$B_i = \frac{K}{r_i^3}\sqrt{1 + 3\sin^2(\theta_i)} \quad i = 1, 2, ..., n. \tag{6.2}$$

In this context, $K = \frac{\mu_0 N_t I F}{4\pi}$, where N_t describes the number of turns of the wire, I is the current running through the coil, F expresses the base area of the coil, μ_0 is the permeability of free space, r_i is the distance between the MS and coil i, and θ_i is the MS elevation angle relative to the coil plane [32, 78]. Figure 6.11 illustrates the elevation angle of the MS in respect to the coil plan.

In the following subsections, we describe three methods to calculate the position of the MS: The first one is the algebraic multilateration method, the second and third methods are the NLS-based methods, which use the estimated position from the first method as a start value. Since the algebraic multilateration algorithm does not always deliver an optimized position due to the nonlinear measurement model, we used the NLS Gauss–Newton as well as the Levenberg–Marquardt algorithm. Furthermore, we derived the equations to compute these NLS methods in a convenient form for MCUs.

6.6.1. Algebraic Multilateration Method

In the two-dimensional case (2D), when the coil i and the magnetometer lay on the same horizontal plane, θ_i is equal to zero. Thus, (6.2) is reduced to the following equation:

$$B_i = \frac{K}{r_i^3} \quad i = 1, 2, ..., n. \tag{6.3}$$

The position of the MS is computed by using the algebraic multilateration method [168], based on the distances to n coils, calculated according to:

$$r_i = \sqrt[3]{\frac{K}{B_i^3}} \quad i = 1, 2, ..., n. \tag{6.4}$$

In the general three-dimensional case (3D), the unknown elevation angles θ_i can be estimated by using a three-axis accelerometer, which enables the measurement of the pitch angle β and the roll angle κ of the MS [32]. Based on the measured pitch and roll angles, the elevation angles between the MS and reference stations can be calculated as follows:

$$\theta_i = \arctan\left[-\frac{3}{4}\tan I_i \pm \sqrt{\left(\frac{3}{4}\tan I_i\right)^2 + \frac{1}{2}}\right], \tag{6.5}$$

whereby I_i is the inclination of the magnetic field from coil i, which is calculated using:

$$I_i = \arcsin\left[\frac{-B_{x',i}\sin\beta + B_{y',i}\cos\beta\sin\kappa + B_{z',i}\cos\beta\cos\kappa}{B_i}\right], \tag{6.6}$$

where $B_{x',i}$, $B_{y',i}$ and $B_{z',i}$ are the magnetic field components in the coordinate system of the sensor, which is integrated in the MS. Furthermore, $B_i = \sqrt{B_{x',i}^2 + B_{y',i}^2 + B_{z',i}^2}$, where B_i is the magnetic field magnitude. Hence, the distances r_i between the MS and the reference stations can be calculated based on the estimated elevation angles θ_i:

$$r_i = \sqrt[3]{\frac{K\sqrt{1 + 3\sin^2(\theta_i)}}{B_i}} \quad i = 1, 2, ..., n. \tag{6.7}$$

In this case, the position of the MS can also be calculated by using the algebraic multilateration algorithm. Therefore, like the UWB-based localization system, the pseudo-inverse matrix can be processed in a computing unit such as a PC or a laptop, in order to initialize the MCU with the preprocessed result. In this way, the MCU has only to compute a matrix multiplication to estimate an MS's position.

6.6.2. Non-linear Least Squares Method: Gauss–Newton Algorithm

Based on (6.2), the (x, y, z) coordinates of the MS can be computed by solving the following nonlinear system of equations:

$$f_i(x, y, z) = \frac{K}{r_i^3}\sqrt{1 + 3\sin^2(\theta_i)} - B_i = 0 \quad i = 1, 2, ..., n \tag{6.8}$$

whereby, $r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$, $\sin\theta_i = \frac{z - z_i}{r_i}$; (x_i, y_i, z_i) and (x, y, z) are the coordinates of the i-th coil and the MS, respectively. Further, B_i is the measured magnetic field strength.

For simplicity, we set $u = (x, y, z)$, the Gauss–Newton algorithm iteratively finds the best estimate $\hat{u}$, which minimizes the sum of squares:

$$\hat{u} = arg\min_{u} \sum_{i=1}^{n} (f_i(u))^2 \tag{6.9}$$

The Gauss–Newton method starts with an initial guess $\vec{x}^{(1)}$ calculated by the algebraic multilateration method and proceeds iteratively (see Equations 5.10 and 5.11) [169, 197]; where $\boldsymbol{J_f}$ is the Jacobian matrix of the function: $f = (f_1, f_2, ..., f_n)$ at u_k. The Jacobian matrix $\boldsymbol{J_f}$ is calculated as well as simplified in a form suitable for resource-constrained devices by using Equation (6.8) to:

$$\boldsymbol{J_f} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial y} & \frac{\partial f_i}{\partial z} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x} & \frac{\partial f_n}{\partial y} & \frac{\partial f_n}{\partial z} \end{pmatrix}, \quad i = 1, 2, ..., n \tag{6.10}$$

Whereby, $\frac{\partial f_i}{\partial x}$, $\frac{\partial f_i}{\partial y}$, and $\frac{\partial f_i}{\partial z}$ are respectively equal to:

$$\frac{\partial f_i}{\partial x} = -3K\Delta x_i \left[\frac{(\Delta x_i)^2 + (\Delta y_i)^2 + 5(\Delta z_i)^2}{dm_i}\right], \tag{6.11}$$

$$\frac{\partial f_i}{\partial y} = -3K\Delta y_i \left[\frac{(\Delta x_i)^2 + (\Delta y_i)^2 + 5(\Delta z_i)^2}{dm_i}\right], \tag{6.12}$$

and

$$\frac{\partial f_i}{\partial z} = -12K \left[\frac{(\Delta z_i)^3}{dm_i}\right], \tag{6.13}$$

where, $\Delta x_i = x - x_i$, $\Delta y_i = y - y_i$, $\Delta z_i = z - z_i$, and

$$dm_i = \left((\Delta x_i)^2 + (\Delta y_i)^2 + 4(\Delta z_i)^2\right)^{\frac{1}{2}} \left((\Delta x_i)^2 + (\Delta y_i)^2 + (\Delta z_i)^2\right)^3. \tag{6.14}$$

The iteration process stops when the updates become sufficiently small. Furthermore, the initial guess $\vec{x}^{(1)}$ is calculated by using the multilateration algorithm and the distances r_i as described in the first method (see Section 6.6.1). Like the UWB-based ILS, the Gauss-Newton algorithm uses the Moore–Penrose pseudo-inverse algorithm to calculate the error correction in (5.11) as well as to choose the $\vec{x}^{(k)}$ with the minimum error value.

If the elevation angle θ_i is unknown, it can be approximated by replacing the term $\sqrt{1 + 3\sin^2(\theta_i)}$ by "1.5" in (6.7) [78], or by using an iterative algorithm described in [198].

6.6.3. Non-linear Least Squares Method: Levenberg–Marquadt Algorithm

The Levenberg–Marquadt method is also an algorithm for solving the NLS problems that uses the trust-region approach [169]. The advantage of the trust-region strategy is the stability against the rank-deficiency of the Jacobian matrix $\boldsymbol{J_f}$, which is one of the weaknesses of the Gauss–Newton method [169]. Like the Gauss–Newton method, the Levenberg–Marquadt method proceeds iteratively: $\vec{x}^{(k+1)} = \vec{x}^{(k)} + \vec{s}^{(k)}$, whereby the error correction vector is equal to:

$$\vec{s}^{(k)} = -(\boldsymbol{J_f}^{T(k)}\boldsymbol{J_f}^{(k)} + \mu^{2(k)}\boldsymbol{I})^{-1}\boldsymbol{J_f}^{T(k)}\vec{f}^{(k)}, \tag{6.15}$$

where $\boldsymbol{I}$ is the identity matrix and μ is the damping parameter. The error vector $\vec{s}$ is calculated by using the QR-Householder instead of the Moore–Penrose pseudo-inverse algorithm, since it is less computing time consuming and the Levenberg–Marquadt method is robust against the rank-deficiency of the Jacobian matrix $\boldsymbol{J_f}$.

The initial damping-parameter $\mu^{(1)}$ can be calculated based on the matrix $\boldsymbol{A}^{(1)}$:

$$\boldsymbol{A}^{(1)} = \boldsymbol{J_f}^{T(1)}\boldsymbol{J_f}^{(1)}, \tag{6.16}$$

as follows:

$$\mu^{(1)} = \tau \cdot max_i\{a_{ii}^{(1)}\}, \tag{6.17}$$

where $a_{ii}^{(1)}$ are the diagonal elements of the matrix $\boldsymbol{A}^{(1)}$ and τ is chosen by the user. As a rule of thumb, a small value of τ should be chosen (e.g., $\tau = 10^{-6}$), if the initial guess $\vec{x}^{(1)}$ is believed to be a good approximation; otherwise $\tau = 10^{-3}$ or $\tau = 1$ can be used. Furthermore, the value of $\mu^{(k)}$ can be updated based on the gain ratio ϱ [199]:

$$\varrho = \frac{G^{(k)} - G^{(k+1)}}{G^{(k)} - \vec{g}^{T^{(k)}} \vec{g}^{(k)}}, \tag{6.18}$$

$$G^{(k)} = \vec{f}^{T^{(k)}} \vec{f}^{(k)}, \tag{6.19}$$

$$G^{(k+1)} = \vec{f}^{T^{(k+1)}} \vec{f}^{(k+1)}, \tag{6.20}$$

$$\vec{g}^{(k)} = \vec{f}^{(k)} + \boldsymbol{J}_{\boldsymbol{f}}^{(k)} \vec{s}^{(k)}. \tag{6.21}$$

$$\mu^{(k+1)} = \begin{cases} 2 \cdot \mu^{(k)} & \text{if } \varrho \leq \ \beta_0 \\ \dfrac{\mu^{(k)}}{2} & \text{if } \varrho \geq \beta_1 \end{cases}, \tag{6.22}$$

whereby, $0 < \beta_0 < \beta_1 < 1$ (e.g., $\beta_0 = 0.2$, $\beta_1 = 0.8$).

6.7. Synchronization Evaluation

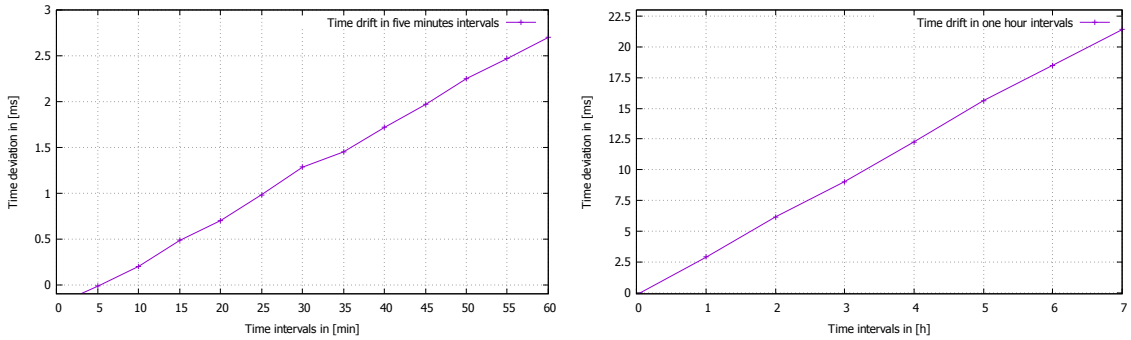
In this section, we present the results from our preliminary experimental evaluation of the accuracy of the DS3234-RTC clocks in a real indoor environment. Subsequently, we validate the correlation of the magnetic field data, which are generated from three coils, with the magnetic values that are acquired from the MS. Furthermore, we investigate the impact of the clock drift on the calculated distances and position using the coil magnetic field measured at the MS.

6.7.1. Clock Drift Evaluation

Firstly, we measure the drift between two DS3234-RTCs by using a Digital Sampling Oscilloscope (DSO) as well as by quantifying the time deviation over a period of three weeks in an average ambient temperature of about 20.5 °C.

According to our measurements, the DS3234-RTCs have an average drift time of about 2.7 ms per hour (0.75 ppm), which corresponds to a 0.4 sample deviation within an hour by the maximal sample rate of 154 sample/s of the HMR2300 magnetometer. Figure 6.12 represents the average drift time within one and seven hours between two DS3234-Clocks, whereby both figures show a time drift, which is different from zero by $t = 0$ s. Although the RTCs are set to the same time by $t = 0$ s, they show an average drift time from one up to two-tenths of milliseconds, due to the time delay during the initialization phase.

According to the measurements of the RTCs, the magnetic field data on the MS would have an average sampling error of 30 samples in three days, which corresponds to ±19% of the coil cluster time (*cf.* Figure 6.2(a)), and will be acknowledged in the results of the second experiment. The observed average drift time of about 2.7 ms does not differ significantly



(a) *The mean clock drifts during* 1 *h.* **(b)** *The mean clock drifts during* 7 *h.*

Figure 6.12.: *Time deviations of the clocks*

from the theoretical drift of about 3.0 ms, which is calculated by using Equation (6.23) [200]:

$$ERR(ppm) = -0.042 * (25 - T)^2, \tag{6.23}$$

whereas ERR is the theoretical crystal frequency error and T is the operating temperature. Theoretically, in the worst case, the average sample error after an hour is 0.4 samples, which can lead to approximately ±20% of each coil cluster signal after three days. The RTC time drifts away from the real time in a different direction and at a different rate (slow or fast), which depends on the factors described in Section 6.4.1 and other subtle environmental variables.

6.7.2. Clock Drift Impact

Secondly, to examine the synchronization on the MS and the impact of the clock drift on the calculated distances and position, we setup a real-world experiment with three CDUs and one MS, which gathers and synchronizes the magnetic data on a fixed position (*cf.* Figure 6.13). The true distances between the MS and coils 1, 2 and 3 are 2.15 m, 3.45 m and 3.9 m, respectively. The coils are placed in the corners of two rectangular rooms with a $6.7 \times 5.3\,m^2$ and $6.7 \times 5.79\,m^2$ surface. The rooms are separated by a 0.58 m thick wall (see Figure 6.13). The three coils are controlled in two different ways, via One-CDU or Three-CDU units to compare the two CDU-configurations with respect to the clock drift as well as the distance and position deviation. Furthermore, the synchronized data and the calculated distances are logged in real-time on the MS, then transferred and subsequently evaluated on a PC. At the beginning of the experiment, the clocks are initiated once. In order to provide the maximum time threshold affecting the distance measures, none of the clocks were resynchronized during the experiment. The distances d_1, d_2, and d_3 between the MS and coil 1, 2, and 3 are respectively calculated for a period of 104 h every four hours.

The vertical red lines in Figures 6.14 and 6.15 represent the beginning of coil 1's first

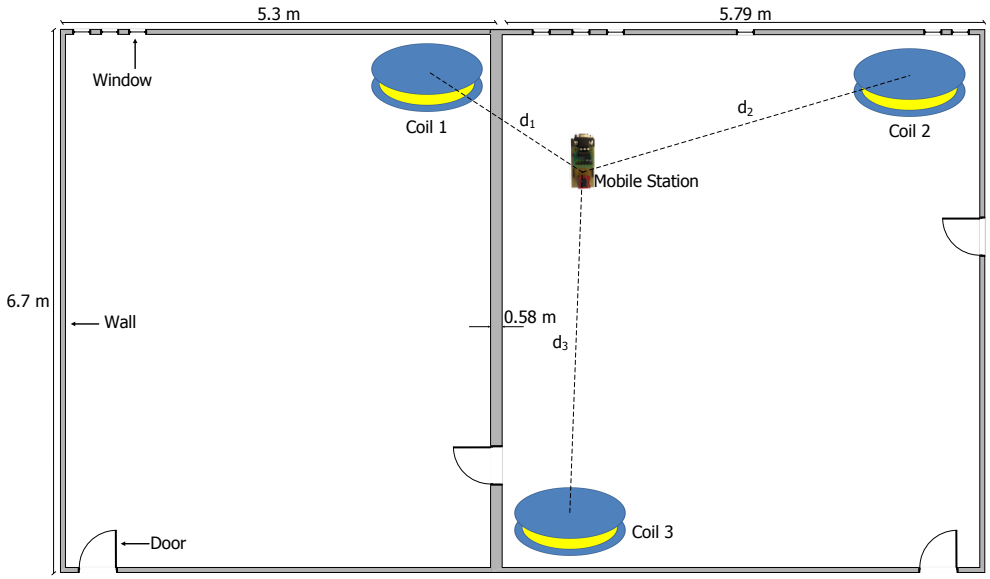
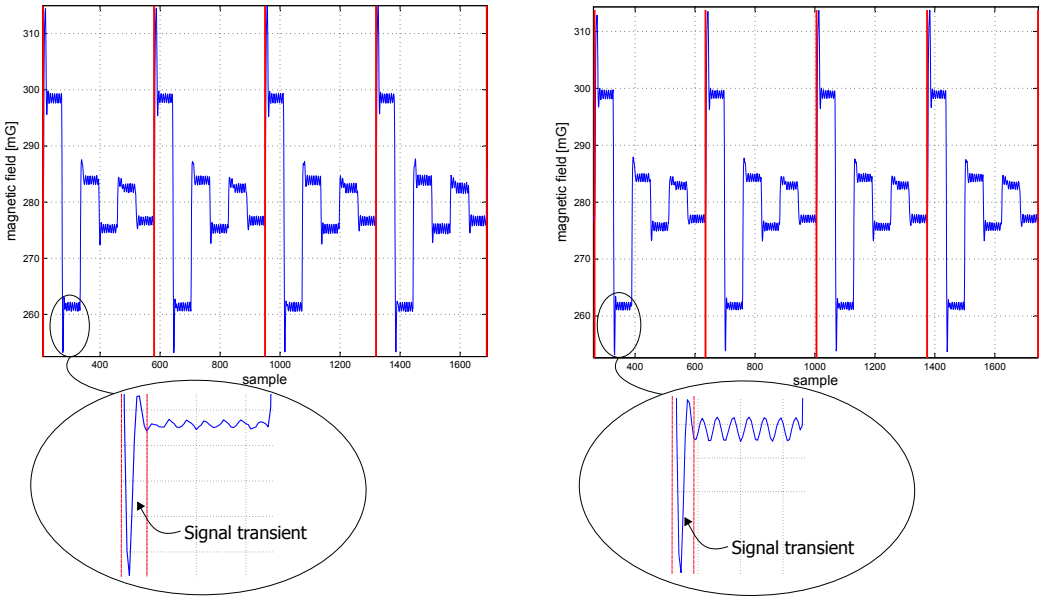


Figure 6.13.: *Experimental setup for distance measures between a MS and three coils*

cluster, which is detected by the MS. Figure 6.14 shows the coil clusters after the initiation of the CDU units and the MS in the One- and Three-CDU configurations. Signal transients occur at the beginning of each cluster, due to the switching of inductive loads (coils).



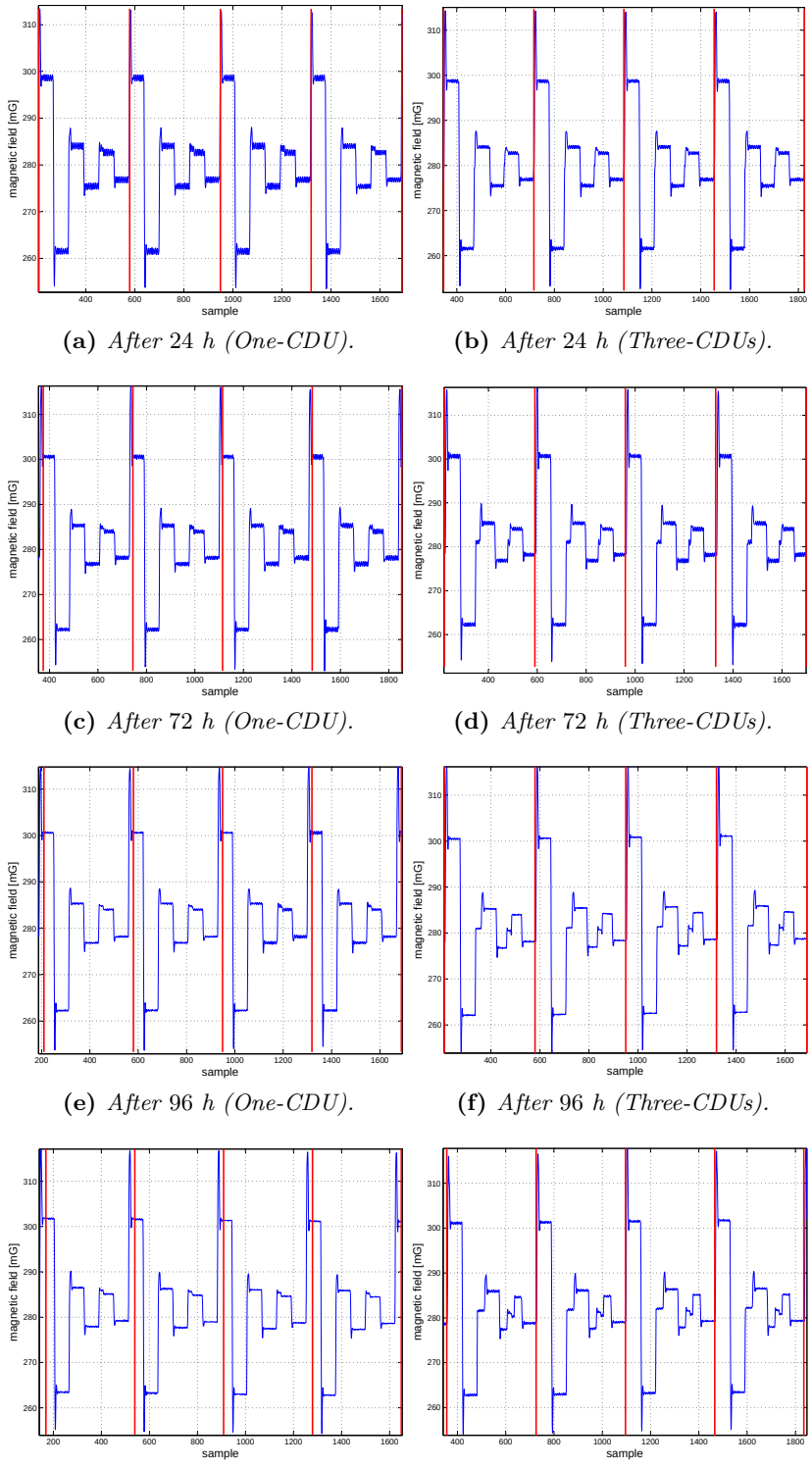
(a) *One-CDU.* **(b)** *Three-CDUs.*

Figure 6.14.: *The magnetic field at the beginning of the experiment and signal transients*

Figure 6.15 shows the coil clusters for both One- and Three-CDU configuration after 24 h, 72 h, 96 h and 120 h. The coil's cluster signals are free of distortion after one day (*cf.* Figures 6.15(a) and (b)). After three days, the sampling signals are distorted by using the Three-CDU configuration (*cf.* Figures 6.15(d), (f), and (h)); in contrast, the signals remain unchanged and distortion free by the One-CDU configuration (*cf.* Figures 6.15(c), (e), and (g)). The signals are free of distortion, since the coils are controlled via one single CDU. By contrast, the signal distortion of the N-CDU configuration is due to the superposition of magnetic fields originating from coils with neighboring time slots (e.g., time slots 1 and 2). A signal superposition occurs when the CDU of a neighboring time slot does not begin or terminate the coil activation at the right time. After 96 h or 120 h, strong signal distortions arise, which are shown in Figures 6.15(f) and (h), respectively. Although the One-CDU configuration does not exhibit any cluster distortion, the time drift between the CDU- and the MS-RTC rises steadily with time and reaches a time drift of about a third of the cluster time after 120 h (*cf.* Figure 6.15(g)). The time drift between the CDU-RTCs and the MS-RTC is marked by the red vertical lines.

The calculated distances d_1, d_2, and d_3 between the coils and the MS remain relatively constant in both CDU-configurations after a time lapse of 72 h and have an error up to a few centimeters (*cf.* Figure 6.16), since the One- and Three-CDU configurations feature small time drifts within this time interval, and the clusters are not strongly distorted in the Three-CDU configuration (*cf.* Figure 6.15(d)). Furthermore, the gathered values of the magnetic field are ignored from the MS for the transient period (*cf.* Figure 6.14), which consequently decreases the effect of the time drift between the RTCs. By the One-CDU configuration, the distances stay relatively stable in the time interval $[72\,h, 92\,h]$, whereby the maximum deviations of the calculated distances d_1, d_2, and d_3 are about 21 cm, 30 cm and 34 cm, respectively (*cf.* Figure 6.16(a)). In contrast, during the same interval, the maximum deviations of the calculated distances d_1, d_2 and d_3 are about 4 cm, 40 cm and 43 cm by the Three-CDU configuration, respectively (*cf.* Figure 6.16(b)). As expected, the N-CDU configuration generally exhibits a larger distance deviation compared against the One-CDU configuration, since the N-CDU configuration is subject to $(N+1)$ error sources: $N-$CDU and one MS RTCs. This leads to a false synchronization between the CDU-RTCs and the MS-RTC, as well as to a superposition between the coils' magnetic fields of successive time slots. Furthermore, due to signal transients, five values of the captured magnetic field data, which correspond to a 40 ms guard time, are discarded from the beginning and the end of each cluster. Thus, the calculated distances and positions have not yet been noticeably affected by time drift. However, particularly after 92 h, the drift is significantly bigger than 40 ms and consequently leads to higher errors in the calculated distance and positions.

The distance d_1 in the Three-CDU configuration remains virtually stable after a time lapse of about 104 h (*cf.* Figure 6.16(b)); it presents an exception, since the MS-RTC and the CDU-RTC of the first coil coincidentally drift in the same direction and with nearly



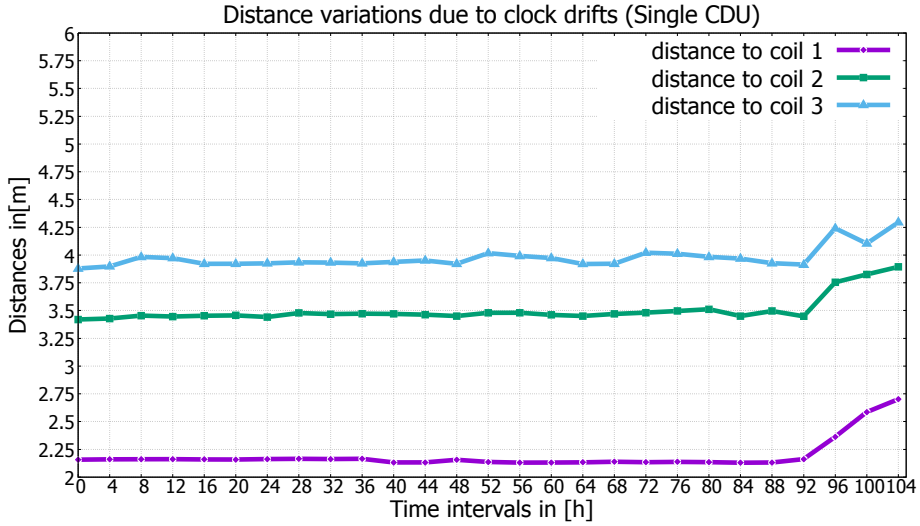
(a) *After* 24 *h (One-CDU).* **(b)** *After* 24 *h (Three-CDUs).*

(c) *After* 72 *h (One-CDU).* **(d)** *After* 72 *h (Three-CDUs).*

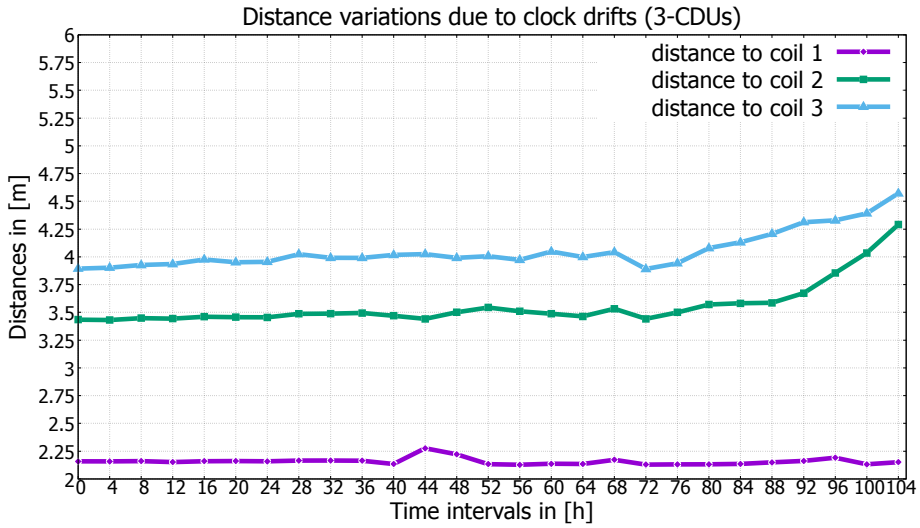
(e) *After* 96 *h (One-CDU).* **(f)** *After* 96 *h (Three-CDUs).*

(g) *After* 120 *h (One-CDU).* **(h)** *After* 120 *h (Three-CDUs).*

Figure 6.15.: *The magnetic field samples of three coils (One-CDU and Three-CDUs)*



(a) *One-CDU configuration.*

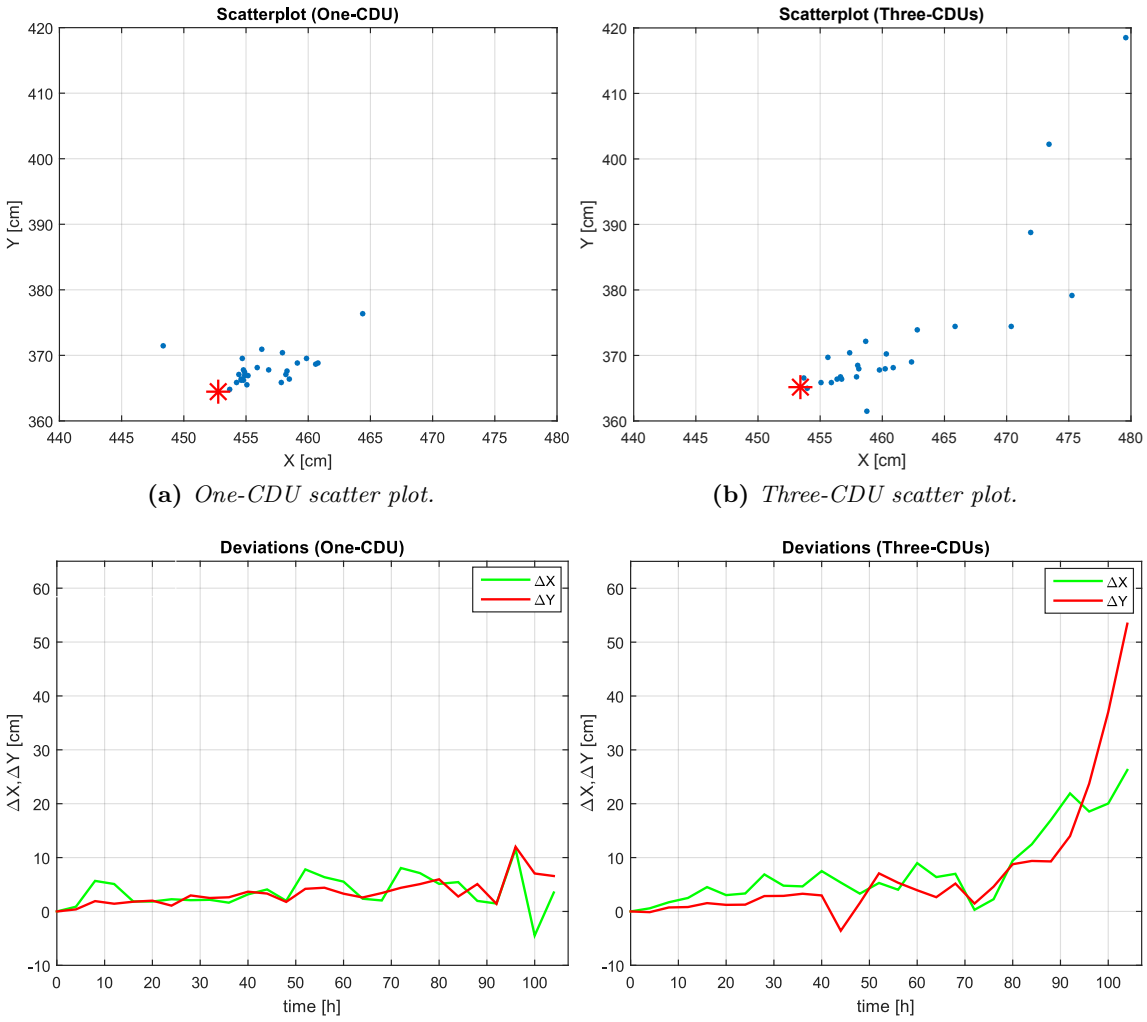


(b) *Three-CDU configuration.*

Figure 6.16.: *Comparison of distance deviations*

equal rate. Therefore, there is no time drift between the MS-RTC and the first CDU-RTC (see the vertical red lines in Figures 6.15(b), (d), (f), and (h)). Moreover, the first cluster remains distortion free; since, by coincidence, the RTC of the third cluster (the third time slot) ceases early and the RTC of the second cluster begins late to deactivate or activate the coils.

The MS position is determined by using the distances d_1, d_2, and d_3 from the MS to the coils. The distances are calculated on the MS and transmitted in real-time to a PC to compute the MS position deviations in regard to the time drift for both CDU-configurations. Figure 6.17 shows the position scatter as well as deviations of the MS for the One- and Three-CDU configurations, whereby the One-CDU demonstrates less variation than the



(a) *One-CDU scatter plot.* **(b)** *Three-CDU scatter plot.*

(c) *One-CDU position deviations.* **(d)** *Three-CDU position deviations.*

Figure 6.17.: *Comparison of position scatter and deviations*

Three-CDUs in relation to the calculated position (*cf.* Figures 6.17(a) and (b)).

The Three-CDU configuration exhibits more variance than the One-CDU configuration in the x- and y-components of the MS coordinates after a time lapse of about 80 h (see Figures 6.17(c) and (d)), since this configuration depicts more distance deviations due to cluster distortions and the time drift to the MS compared to the One-CDU configuration. The Three-CDU configuration causes more distance deviation than the One-CDU configuration and, hence, diminishes the precision of the calculated position, particularly after a time period of about 80 h.

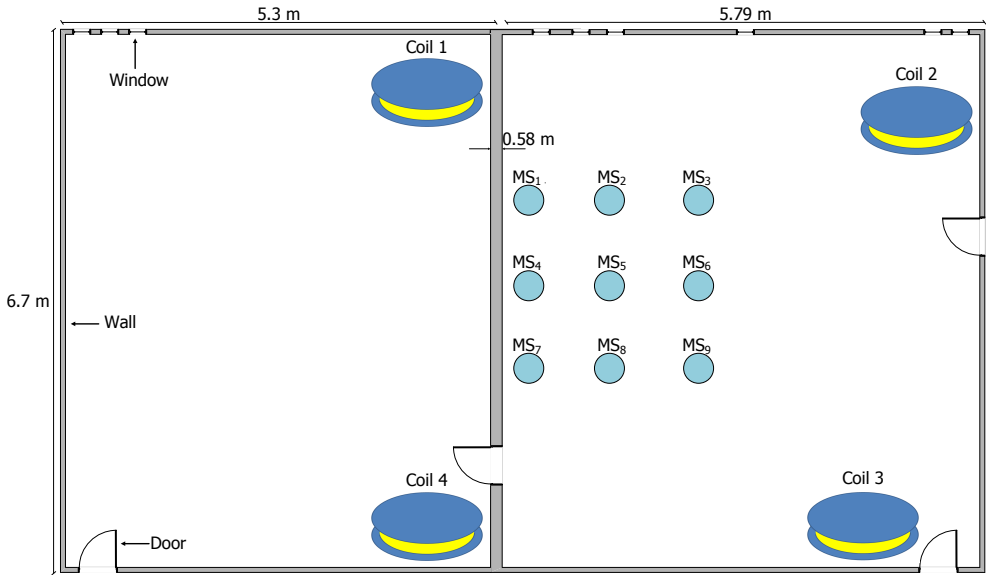


Figure 6.18.: *Experimental setup for position measures between various MSs and three coils*

6.8. System Evaluation

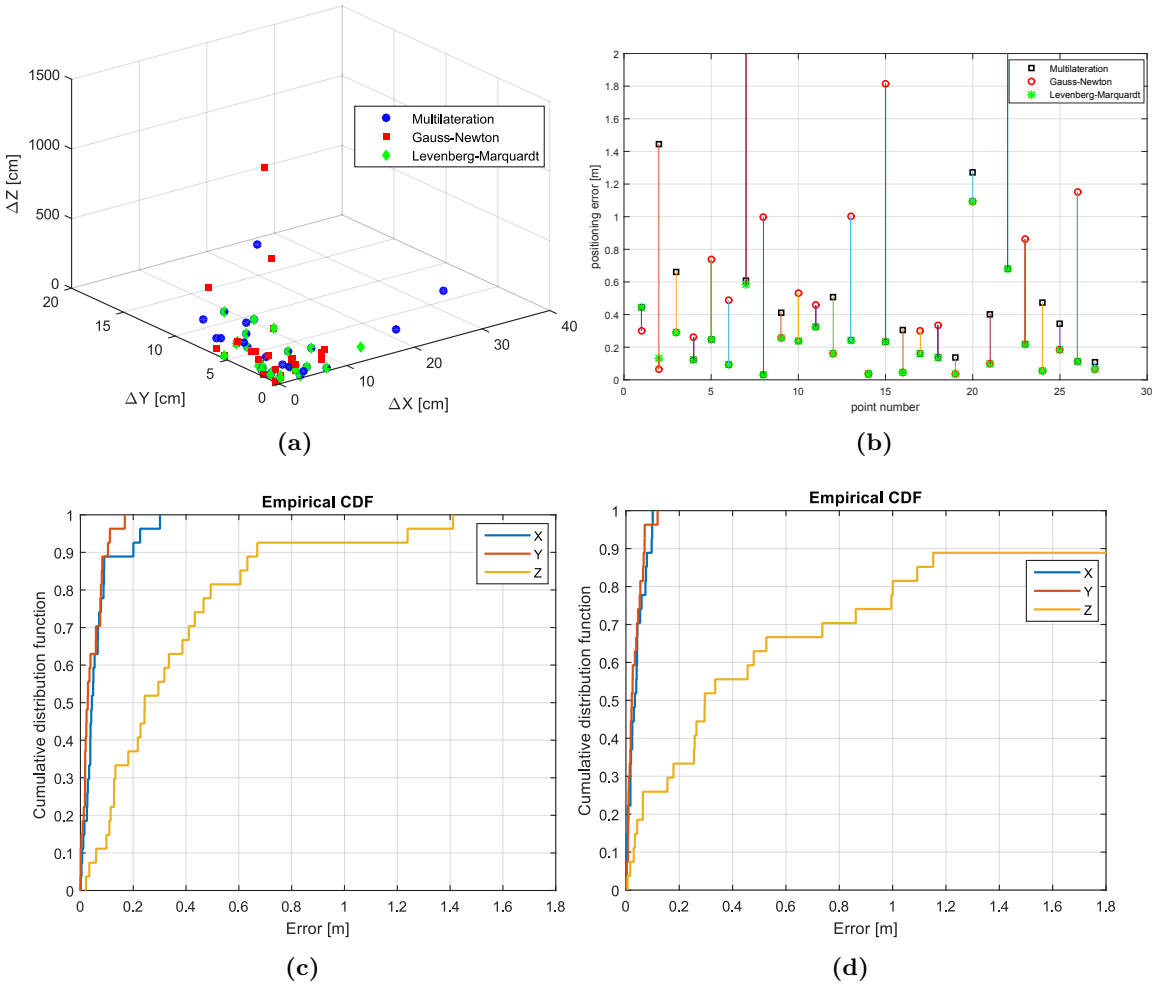
In this section, we present the results of the experimental evaluation of the presented MILPS. The aim of this evaluation is to demonstrate the feasibility to implement the proposed platform of MILPS. Therefore, as in the previous experiment in Chapter 5, we will not address issues such as the impact of the placement of the anchors or the location of the MS, as well as the selection of the anchors on the localization accuracy. These issues are treated in Chapter 7. The complexity of the algorithms used is presented in Section 5.3. For the evaluation, we give the results of the accuracy measurements of the MILPS. Furthermore, we evaluated the computing time of the algorithms on the LPC2387-MCUs, which are running at the 72 MHz. Finally, we evaluated the energy consumption of the algorithms as well as of the MS by MILPS in Section 6.8.3.

For the computing time and the accuracy evaluation of the MILPS, we placed four coils inside two rectangular rooms. The two rooms are separated by a 0.58 m thick wall (see Figure 6.18). Two coils are placed in a room with a surface of $5.3 \times 6.7\,m^2$, while the other coils are placed in a room with a surface of $5.79 \times 6.7\,m^2$. The MSs are placed in a variety of twenty-seven positions. The MSs represented in Figure 6.18 are in three different heights of 0.655 m, 1.423 m, and 2.308 m.

The true positions of the reference and mobile stations are determined by geodetic methods with millimeter accuracy using a tachymeter. We choose this configuration to demonstrate that the MILPS can measure the position even if the coils and the MSs are separated by walls.

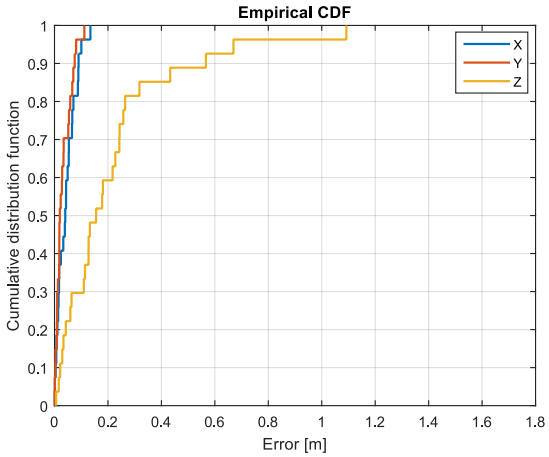
6.8.1. Positioning Accuracy Measurement

Figure 6.19 presents a comparison between the algebraic multilateration, Gauss–Newton and Levenberg–Marquardt methods in terms of errors, which are illustrated in Figure 6.19(a). Figure 6.19(b) shows the positioning error, which is defined as the Euclidean distance between the estimated and true position. As illustrated by this figure, the position errors of point numbers 7 and 22 are out of the bound by the Gauss–Newton algorithm, since it diverges. Figure 6.19(c) shows the experimental results of the positioning errors obtained by the algebraic multilateration method. The positioning errors in Figure 6.19(c) are represented by the empirical CDF. In this experiment, the error in the x and y components is lower than 30 cm. However, the z component of the MS coordinates shows the worst performance, since three coils were placed at nearly equal heights.



(a) (b) (c) (d)

Figure 6.19.: *Cont.*



(e)

Figure 6.19.: *MILPS: scatter plots, position errors, and empirical CDFs. GNM, Gauss–Newton Method; ML, Multilateration; LVM, Levenberg–Marquardt. (**a**) Scatter plot of the ML, the GN, and the LVM algorithms, (**b**) position error of the ML, the GN, and the LVM algorithms, (**c**) empirical CDF of the points estimated by the ML algorithm, (**d**) empirical CDF of the estimated positions after the use of GNM, (**e**) empirical CDF of the estimated positions after the use of LVM algorithm.*

The application of the Gauss–Newton method reduced the errors in the x- and y-coordinates to 10 cm but impaired the z-coordinate values (see Figure 6.19(d)). In contrast, by using the Levenberg–Marquardt method, the errors in the z-coordinates are limited to 0.6 m for 93 % of the measured points and to 1.1 m for other points (see Figure 6.19(e)). Furthermore, Figure 6.19(a) shows that the Levenberg–Marquardt method has generally lower deviations in all coordinate components compared with other methods.

6.8.2. Computing Time Measurement

As explained in Subsection 6.6.1, we used the multilateration method to estimate the position of the MSs illustrated in Figure 6.18. Therefore, the matrix $\boldsymbol{A}^{+}$ is also calculated based on the Moore–Penrose method, only once, at the start of the MCU. Like the UWB-based system, the position estimation of the MSs can be optimized by using the Gauss–Newton or the Levenberg–Marquardt algorithms. Both algorithms utilize the position carried out by the algebraic multilateration method as a starting point and proceed iteratively up to the desired accuracy or when the maximal iteration number is reached. The average iteration number in this experiment is seven and six for the Gauss–Newton and the Levenberg–Marquardt method, respectively. The mean time of calculating a starting position by resource-constrained LPC2387-MCU is approximately 0.1 ms. The computing time of an estimated position increases approximately up to 33 ms or 21 ms by using the Gauss–Newton

or the Levenberg–Marquardt algorithm, respectively. The evaluation of the described computing steps and the NLS methods used is summarized in Table 6.3.

Table 6.3.: *Mean computing times of the algorithms used by MILPS. Computing times measured on an LPC2387 running at 72 MHz*

Algorithm	**Computing time** (µs)
$\boldsymbol{A}^+$ for the multilateration method (at the start)	4563
Multilateration method	92
Gauss–Newton method per iteration	4645
Levenberg–Marquardt method per iteration	3467

6.8.3. Energy Consumption

The energy consumption of the algorithms is measured based on the measurement of the drain-source current in the supply line, which is powered by a reference voltage supply V_{cc} ($V_{cc} = 5\,\mathrm{V}$). Hence, the energy used for each localization processing task can be calculated by integrating the electric power over the times, which are summarized in Table 6.3. We measured a current consumption of about 69 mA at the ambient temperature of 26 $^\circ C$ for the LPC2387-MCU in the active mode. The measured energy for the localization algorithms by MILPS is summarized in Table 6.4.

Table 6.4.: *List of measured energy consumption values of the algorithms by the MILPS*

Algorithm	**MILPS Energy** (µWs)
$\boldsymbol{A}^+$ by the multilateration method (at the start)	1574.24
Multilateration method	31.74
Gauss–Newton per iteration	1602.53
Levenberg–Marquardt method per iteration	1196.12

We also measured the energy consumption of the MS performed for the MILPS by using the method for the energy consumption of the algorithms. We measured a current consumption of about 27 mA by the magnetometer. Whereas, the measurement time by the MILPS is 1 s. The total energy, which is required for a position estimation, is calculated based on the current drain of the magnetometer sensor as well as the energy consumption of the MCU (see Table 6.4). The energy usage of the MILPS is summarized in Table 6.5, whereby the energy consumption of the magnetometer is 405 mW.

Table 6.5.: *List of measured energy consumption of MILPS for a position estimation*

Optimization algorithm	**Energy [mWs]**
MILPS (Gauss–Newton)	$405 + 1.6 + 0.032 + 7 \times 1.63 \simeq 418.04$
MILPS (Levenberg–Marquardt)	$405 + 1.6 + 0.032 + 6 \times 1.21 \simeq 413.89$

6.9. Conclusion

In this chapter, we present a platform for an indoor location system that is designed and implemented for a decentralized architecture using magnetic field strength as a measurement method. This magnetic-based localization system can be used for specific applications such as for the beverage industry to monitor processes in liquid-filled containers or in harsh environments such as a coal mine. The magnetic-based system can compensate the RF technology in the case of materials such as copper, steel or liquid due to high attenuation. It can replace light-wave technology with opaque materials such as muddy liquid, dust or smoke as well as being used instead of acoustic wave technology because of effects like scattering or reflections. The magnetic-based ILS can also serve as an extension for the UWB-based ILS to cover areas where the UWB signal is not available.

Although magnetic-based ILSs enable localization in NLoS scenarios as well as in opaque environments, they provide a limited coverage area. A wide coverage requires a high energy consumption as well as large coils. Magnetic localization systems that are based on very low frequencies or static magnetic fields are sensitive to high magnetic noise sources such as electric equipment and power lines [201]. Furthermore, the measured signals are distorted by ferromagnetic and magnetic objects near the magnetometer [202].

CHAPTER 7

Algorithms and Position Optimization for Resource-Constrained Devices

In this chapter, we describe suitable algorithms to compute a start position. These algorithms will be analyzed with respect to stability, complexity, and memory requirements. The accuracy of the calculated position depends on the location of the reference points as well as of the position of the searched point relative to the reference stations. A metric about the relative position of the reference station to each other and to the unknown position is the Dilution of Precision (DOP). DOP can be expressed more specifically by separate measurements [203]:

- Horizontal Dilution of Precision (HDOP): Gives a measure about the horizontal positioning accuracy.
- Vertical Dilution of Precision (VDOP): Gives a measure about the vertical positioning accuracy.
- Time Dilution of Precision (TDOP): Gives a measure about the timing accuracy.
- Geometric Dilution of Precision (GDOP): Includes HDOP, VDOP, and TDOP.

We use PDOP, which is a measure about the expected position inaccuracies in the x-, y-, and z-directions. Furthermore, the accuracy of the position estimate also depends on the accuracy of the sensors which, for example, measure a distance or magnetic field strength to/from the reference stations. Therefore, the calculated start position can be optimized by using optimization algorithms such as the Gauss–Newton or the Levenberg–Marquardt methods.

We analyze and compare the GNM with two variants of the LVM methods. The LVM methods used are the Dahmen and Reusken LVM (DR-LVM) and Madsen LVM (M-LVM) [199, 204]. Experience and the analysis show that the use of the optimization algorithms is not always necessary. Therefore, we develop an adaptive algorithm for the optimization of the position, which is based on the SVD, LVM algorithm, and the PDOP. This method allows for an adaptive selection mechanism for the LVM algorithm. This adaptive algorithm enables saving of resources such as memory, computing time, and energy on resource-constrained devices. Furthermore, the parameters of the LVM algorithm impact the accuracy as well as the required iteration numbers. Therefore, the influence and the choice of the right

parameter combination will be analyzed and discussed. In addition, we design and evaluate a method to detect and mitigate multipath effects on the mobile station, which enables an accurate localization in non-line of sight scenarios. This method is implemented and evaluated in simulated and real environments [15].

7.1. Primary Problem

The trilateration is a localization algorithm, which computes the position of an unknown MS by measuring its distance from three reference positions. The unknown position can also be calculated with the multilateration method, if more than three reference points are accessible. Assume (x, y, z) and (x_i, y_i, z_i) for $i = 1, 2, ..., n$, are the coordinates of the MS and of n reference points, respectively. In addition, the measured distances between the reference points and the MS are d_i, which are the radii of the individual spheres. The searched location of the MS is the intersection of the spheres, whose equations are:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = d_i^2 \qquad i = 1...n \tag{7.1}$$

The system of nonlinear equations in (7.1) can be solved by different methods [165, 166, 167]. Most methods translate the non-linear Equation (7.1) into a linear equation by relating it to a specific reference station. Therefore, we solved (7.1) by transforming the system of equations into a matrix form [168]. The algorithm used is not related to a specific anchor, since other algorithms subtract the coordinates of a specific anchor for the linearization of the equation system. Additionally, the algorithm gives a measure of the solvability of the multilateration problem and provides a recursive least square approach to update the position [168]. The solution of the linearized system is completely determined if the distances from four reference points are known. The problem requires the estimation of the unknown position $\vec{x} = (x, y, z)$ such that:

$$\boldsymbol{A}\vec{u} = \vec{b}, \tag{7.2}$$

where $\vec{u} = (x^2 + y^2 + z^2, x, y, z)$, the matrix $\boldsymbol{A}$ and the vector $\vec{b}$ have the following forms [168]:

$$\boldsymbol{A} = \begin{bmatrix} 1 & -2x_1 & -2y_1 & -2z_1 \\ 1 & -2x_2 & -2y_2 & -2z_2 \\ 1 & -2x_3 & -2y_3 & -2z_3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & -2x_n & -2y_n & -2z_n \end{bmatrix}, \tag{7.3}$$

$$\vec{b} = \begin{bmatrix} d_1^2 - x_1^2 - y_1^2 - z_1^2 \\ d_2^2 - x_2^2 - y_2^2 - z_2^2 \\ d_3^2 - x_3^2 - y_3^2 - z_3^2 \\ \vdots \\ d_n^2 - x_n^2 - y_n^2 - z_n^2 \end{bmatrix}. \tag{7.4}$$

The solution of Equation (7.2) represents the primary problem for a position estimation by using distances to the reference stations. This can also be solved in the case of the MILPS, which is a signal-based localization system if the elevation angles to the reference stations are known. Since the aim is the implementation of a localization system based on resource-constrained devices, it is important to use stable, efficient and resource saving algorithms to solve the primary Equation (7.2).

Finally, the position obtained from Equation (7.2) can be optimized, since it depends on various factors such as the measurement inaccuracies, the location to the reference points, the non-linearity of the measurement model. The requested accuracy can be application-dependent. For example, determining if a person or an object is in a room requires low location accuracy, which can measure a couple of meters. Depending on the accuracy of the sensors as well as the algorithms used, the solution of (7.2) can reach an accuracy from meter to centimeter range (*cf.* Section 7.6). The position obtained from Equation (7.2) can be optimized by using optimization algorithms such as the Gauss–Newton or the Levenberg–Marquardt. The position calculated from (7.2) represents a starting solution for the optimization algorithm, which can reach an accuracy up to a few millimeters (*cf.* Section 7.5 and 7.6).

7.2. Finding of an Approximate Position

The positioning problem in Section 7.1 is described using the matrix model in (7.2). Solving the system of linear equations in (7.2) is a central field of matrix computations, whereby various decompositions are applied to implement efficient matrix algorithms. Four important decompositions of the matrix $\boldsymbol{A}$ are: the Cholesky (see Section 7.2.1), LU (Lower Upper, see Section 7.2.2), QR (see Section 7.2.3), and the SVD (Singular-Value Decomposition, see Section 7.2.6) decomposition, whereas $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$, $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$, $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$, and $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, respectively. In this work, the Cholesky factorization is used in combination with the normal equation.

7.2.1. Normal Equation by Cholesky Factorization

The solution of (7.2) can be reduced to solve the linear equation:

$$\boldsymbol{A}^T\boldsymbol{A}\vec{x} = \boldsymbol{A}^T\vec{b}. \tag{7.5}$$

This equation is called the normal equation, which has some numerical limitations. Vital information may be lost during the formation of the matrix $\boldsymbol{A}^T\boldsymbol{A}$ due to the finite precision arithmetic [205]. In addition, the normal equation is more sensitive to perturbations than the original linear system $\boldsymbol{A}\vec{x} = \vec{b}$, since its accuracy depends on the condition number $\kappa\left(\boldsymbol{A}^T\boldsymbol{A}\right)$: $\kappa\left(\boldsymbol{A}^T\boldsymbol{A}\right) = [\kappa(\boldsymbol{A})]^2$, where $\kappa(\boldsymbol{A})$ is the condition number of the matrix $\boldsymbol{A}$. Therefore, the normal equation can lose twice as many digits of accuracy as approaches based on QR factorization or SVD [206].

The Cholesky decomposition can be used to solve the normal equation, since $\boldsymbol{A}^T\boldsymbol{A}$ is symmetric and positive definite [206]. The total cost of calculating $\boldsymbol{A}^T\boldsymbol{A}$, $\boldsymbol{A}^T\vec{b}$, and the Cholesky factorization is about $n^2m + \frac{1}{3}n^3$ flops [206].

7.2.2. LU Decomposition

The LU decomposition factorizes the matrix $\boldsymbol{A}$ in a lower triangular matrix $\boldsymbol{L}$ and an upper triangular matrix $\boldsymbol{U}$. The matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ have the entries $\ell_{ij} = 0$ and $u_{ij} = 0$ for all $j > i$ and $i > j$, respectively:

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}. \tag{7.6}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \ell_{21} & 1 & 0 & \dots & 0 \\ \ell_{31} & \ell_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & \ell_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}.$$

Equation (7.2) can be rearranged to:

$$\boldsymbol{L}(\boldsymbol{U}\vec{x}) = \vec{b}, \tag{7.7}$$

whereby:

$$\boldsymbol{L}\vec{y} = \vec{b} \tag{7.8}$$

$$\boldsymbol{U}\vec{x} = \vec{y} \tag{7.9}$$

The linear equation system $\boldsymbol{A}\vec{x} = \vec{b}$ can be solved by first resolving Equation (7.8) then Equation (7.9).

Although the LU decomposition is easy to program as well as needs only about the

half iteration number compared to the QR iteration, it is limited to a square matrix and could easily be unstable [207]. The restriction to square matrices means that the LU decomposition can be used only for a system with four reference stations, which is not practical for localization applications. Hence, we focus only on the QR and the SVD decomposition due to the numerical instability as well as the restriction to the quadratic matrices of the LU factorization.

7.2.3. QR Decomposition

The QR decomposition splits the matrix $\boldsymbol{A}$ into an orthogonal matrix $\boldsymbol{Q}$ and an upper triangular matrix $\boldsymbol{R}$:

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}. \tag{7.10}$$

The matrix $\boldsymbol{Q} \in \mathbb{R}^{mxn}$ is orthogonal if:

$$\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}, \tag{7.11}$$

whereby $\boldsymbol{I}$ is the identity matrix. An orthogonal matrix $\boldsymbol{Q}$ has the following features:

- The column vectors of $\boldsymbol{Q}$ build an orthonormal basis of the vector space $\mathbb{R}^m$.
- The inverse matrix of $\boldsymbol{Q}$ is easy to calculate:

$$\boldsymbol{Q}^{-1} = \boldsymbol{Q}^T. \tag{7.12}$$

The linear system $\boldsymbol{A}\vec{x} = \vec{b}$ can be solved as follows:

$$\boldsymbol{Q}\boldsymbol{R}\vec{x} = \vec{b} \tag{7.13}$$

$$\boldsymbol{R}\vec{x} = \boldsymbol{Q}^{-1}\vec{b} = \boldsymbol{Q}^T\vec{b}, \tag{7.14}$$

whereby $\boldsymbol{Q}^T\vec{b}$ is a matrix-vector multiplication and the unknown vector $\vec{x}$ can be solved by back substitution.

The QR decomposition can be done by applying the Gram-Schmidt, Givens rotation, or Householder algorithm. The Gram-Schmidt method is not recommended in practice, since it is sensitive to rounding errors (numerically unstable) [208]. Therefore, we focus only on the Givens rotation and Householder methods in Sections 7.2.4 and 7.2.5, whereby the Householder approach is the most efficient one [199]. Furthermore, the QR factorization enables to solve overdetermined equation systems, which have more equations than unknowns. That means the row number (m) is greater than of the column number (n) of the matrix $\boldsymbol{A}$. In our case related to localization, the number of the anchors is greater than four.

7.2.4. Givens Rotation Algorithm

The Givens rotation method is based on bringing the column vectors of the matrix $\boldsymbol{A}$ in the vertical position in the direction of the axes by plane rotations to eliminate the corresponding entries. Plane rotations can be described as in Sections 7.2.4.1 through 7.2.4.3.

7.2.4.1. Basic Operation

The basic operation is the rotation of vectors in anticlockwise by an angle θ. Figure 7.1 illustrates the rotation of two unit vectors $\vec{e_1}$ and $\vec{e_2}$ to the vectors $\vec{v_1}$ and $\vec{v_2}$, respectively:

$$\vec{v_1} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \quad \vec{v_2} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

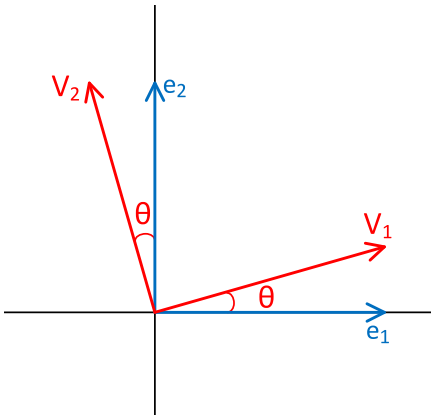


Figure 7.1.: *Basic operation of the Givens algorithm: vector rotation*

The rotation of the vectors $\vec{e_1}$ and $\vec{e_2}$ can be written in matrix form as:

$$G\begin{bmatrix} \vec{e_1} & \vec{e_2} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \boldsymbol{R_\theta}, \tag{7.15}$$

whereas $\boldsymbol{R_\theta}$ is a two-dimensional rotation matrix. A general vector $\vec{x}$ can be expressed as:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1\vec{e_1} + x_2\vec{e_2}$$

The vector $\vec{x}$ can be rotated as:

$$G[\vec{x}] = x_1 G[\vec{e_1}] + x_2 G[\vec{e_2}].$$

Hence, $\vec{x}$ rotates anticlockwise by θ as its components do.

7.2.4.2. Givens Rotation in 2-D

Let $\boldsymbol{A}$ be a general (2×2) matrix: $\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$.

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} \Longleftrightarrow \boldsymbol{Q}^T\boldsymbol{A} = \boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix}. \tag{7.16}$$

This can be achieved by using the Givens rotation that rotates the vector $\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}$ around the z-axis to the vector $\begin{bmatrix} r_{11} \\ 0 \end{bmatrix}$, as illustrated in Figure 7.2. The vector $\begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix}$ will be also rotated, whereby (7.16) can be described as:

$$\begin{bmatrix} c & -c \\ s & s \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix}. \tag{7.17}$$

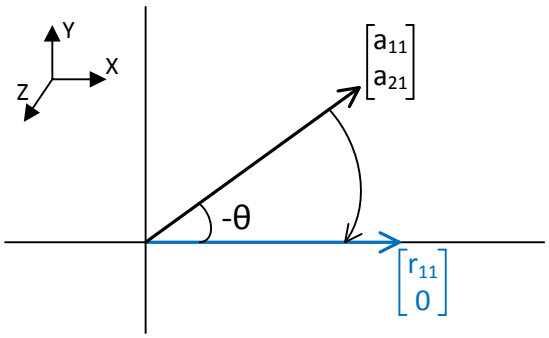


Figure 7.2.: *A 2-D Givens rotation*

$$\cos(-\theta) = c = \frac{a_{11}}{\sqrt{a_{11}^2 + a_{21}^2}}, \ \sin(-\theta) = -s = \frac{a_{21}}{\sqrt{a_{11}^2 + a_{21}^2}},$$

Hence, $\boldsymbol{Q}^T$ is defined as:

$$\boldsymbol{Q}^T = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \frac{1}{\sqrt{a_{11}^2 + a_{21}^2}} \begin{bmatrix} a_{11} & a_{12} \\ -a_{21} & a_{22} \end{bmatrix} \tag{7.18}$$

In this case, the Givens rotation matrix $(\boldsymbol{G})$ is equal to: $\boldsymbol{G} = \boldsymbol{Q}^T$.

7.2.4.3. Givens Rotation in n-D

The Givens rotation matrices are obtained by embedding the previous planar rotations defined in (7.18) in a $(m \times m)$ matrices, which are defined as follows:

$$\boldsymbol{G_{i,j}} = \begin{array}{c} \\ \\ \\ \\ i\rightarrow \\ \\ \\ \\ j\rightarrow \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccc} & & & i\downarrow & & & j\downarrow & & \end{array} \\ \begin{bmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & c & 0 & \dots & 0 & s & \\ & & & 0 & 1 & & & 0 & \\ & & & \vdots & & \ddots & & \vdots & \\ & & & 0 & & & 1 & 0 & \\ & & & -s & 0 & \dots & 0 & c & \\ & & & & & & & & 1 \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{bmatrix} \end{array}, \tag{7.19}$$

whereas $\boldsymbol{G_{i,j}}$ is the rotation in the plan spanned by the unit vectors $\vec{e_i}$ and $\vec{e_j}$ [199, 197]. This rotation affects only the rows i and j.

7.2.4.4. Givens Rotation Principle

This example demonstrates the principle as well as the changes in the matrix illustrated below by using Givens rotations [199]:

$$\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \xrightarrow{G_{1,2}} \begin{bmatrix} \circledast & \circledast & \circledast \\ 0 & \circledast & \circledast \\ * & * & * \\ * & * & * \end{bmatrix} \xrightarrow{G_{1,3}} \begin{bmatrix} \circledast & \circledast & \circledast \\ 0 & * & * \\ 0 & \circledast & \circledast \\ * & * & * \end{bmatrix} \xrightarrow{G_{2,3}} \begin{bmatrix} * & * & * \\ 0 & \circledast & \circledast \\ 0 & 0 & \circledast \\ * & * & * \end{bmatrix}$$

$$\xrightarrow{G_{1,4}} \begin{bmatrix} \circledast & \circledast & \circledast \\ 0 & * & * \\ 0 & 0 & * \\ 0 & \circledast & \circledast \end{bmatrix} \xrightarrow{G_{2,4}} \begin{bmatrix} * & * & * \\ 0 & \circledast & \circledast \\ 0 & 0 & * \\ 0 & 0 & \circledast \end{bmatrix} \xrightarrow{G_{3,4}} \begin{bmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & \circledast \\ 0 & 0 & 0 \end{bmatrix}$$

The new computed matrix entries using $\boldsymbol{G_{i,j}}$ are marked with $\circledast$. The following transformation sequence would also be possible: $G_{1,2}$, $G_{1,3}$, $G_{1,4}$, $G_{2,3}$, $G_{2,4}$, $G_{3,4}$.

7.2.4.5. Givens Rotation Example

$$\begin{bmatrix} 3 & 5 \\ 0 & 2 \\ 0 & 0 \\ 4 & 5 \end{bmatrix} \xrightarrow{G_{1,4}} \begin{bmatrix} 5 & 7 \\ 0 & 2 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \xrightarrow{G_{2,4}} \begin{bmatrix} 5 & 7 \\ 0 & \sqrt{5} \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

where

$$\boldsymbol{G_{1,4}} = \begin{bmatrix} \frac{3}{5} & 0 & 0 & \frac{4}{5} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{-4}{5} & 0 & 0 & \frac{3}{5} \end{bmatrix}, \quad \boldsymbol{G_{2,4}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{2}{\sqrt{5}} & 0 & \frac{-1}{\sqrt{5}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{5}} & 0 & \frac{2}{\sqrt{5}} \end{bmatrix}$$

7.2.5. Householder Algorithm

The Householder approach works with plane reflections on $(n-1)$-dimensional hyperplanes inside of plane rotations. The required zeros in the matrix $\boldsymbol{R}$ are produced by a series of reflections in the right planes. The hyperplanes can be defined by their normal vectors $\vec{v}$, whereby the plan reflections are defined as follows in Section 7.2.5.1.

7.2.5.1. Basic Operation

The Householder method can be established by using projections and reflections, whereby the basic operation is the reflection of a vector at a plane, as illustrated in Figure 7.3.

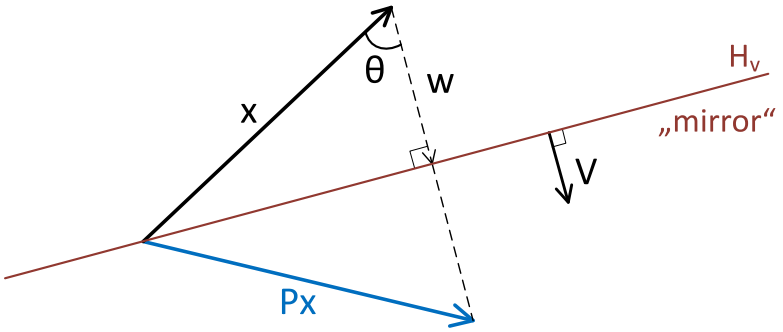


Figure 7.3.: *Basic operation of the Householder algorithm: vector reflection*

Let $\vec{x}$ be a vector and H_v be a plane defined by the normal vector $\vec{v}$, whereby the reflection of $\vec{x}$ is the vector $\vec{P_x}$ (see Figure 7.3). The projection of $\vec{x}$ can be calculated based on the vector $\vec{w}$ as follows:

$$\vec{P_x} = \vec{x} + \vec{w} \tag{7.20}$$

The vector $\vec{w}$ can be determined by its magnitude $\|\vec{w}\|$ and its direction, which is parallel to the vector $\vec{v}$. Therefore, the direction of $\vec{w}$ is equal to $\vec{e_w} = \frac{\vec{v}}{\|\vec{v}\|}$ and so $\vec{w}$ is:

$$\vec{w} = \|\vec{w}\| \, \vec{e_w} = \|\vec{w}\| \, \frac{\vec{v}}{\|\vec{v}\|}, \tag{7.21}$$

The magnitude of $\vec{w}$ can be calculated by using the inner product of the vector $-\vec{x}$ and $\vec{v}$:

$$\langle -\vec{x}, \vec{v} \rangle = -\vec{x}^T \vec{v} = \|\vec{x}\| \, \|\vec{v}\| \cos(\theta),$$
$$cos(\theta) = \frac{-\vec{x}^T \vec{v}}{\|\vec{x}\| \, \|\vec{v}\|},$$

in addition:

$$\|\vec{w}\| = cos(\theta) \, \|\vec{x}\| \, ,$$

hence

$$\|\vec{w}\| = \frac{-\vec{x}^T \vec{v}}{\|\vec{v}\|} \tag{7.22}$$

Thus, by substituting (7.22) in (7.21) we get:

$$\vec{w} = \frac{-\vec{x}^T \vec{v} \vec{v}}{\|\vec{v}\| \, \|\vec{v}\|} = \vec{w} = -\vec{v} \frac{\vec{x}^T \vec{v}}{\|\vec{v}\|^2}$$

$$\vec{w} = -\vec{v} \frac{\vec{x}^T \vec{v}}{\vec{v}^T \vec{v}}, \tag{7.23}$$

whereas $\|\vec{v}\| = \sqrt{\vec{v}^T \vec{v}}$.
From (7.23) two points can be extracted:

- The point from any $\vec{x}$ to the plan (mirror): $\vec{w}$.
- The point from any $\vec{x}$ to the reflection: $2\vec{w}$.

The projection vector $\vec{P_x}$ can be calculated by substituting (7.23) in (7.20):

$$\vec{P_x} = \vec{x} - 2\vec{v} \frac{\vec{x}^T \vec{v}}{\vec{v}^T \vec{v}}$$
$$\vec{P_x} = \vec{x} - 2\vec{v} \frac{\vec{v}^T \vec{x}}{\vec{v}^T \vec{v}}$$
$$\vec{P_x} = \vec{x} \left(\boldsymbol{I} - 2 \frac{\vec{v} \vec{v}^T}{\vec{v}^T \vec{v}} \right), \tag{7.24}$$

whereas, the Householder matrix $\boldsymbol{Q_v}$ is:

$$\boldsymbol{Q_v} = \boldsymbol{I} - 2\frac{\vec{v}\vec{v}^T}{\vec{v}^T\vec{v}} \tag{7.25}$$

The Householder matrix reflects the vector $\vec{x}$ at the plan H_v.

7.2.5.2. Householder Transformation

The basic task is to find $\vec{v} \in \mathbb{R}^n$ for $\vec{y} \in \mathbb{R}^n$ in order that:

$$\boldsymbol{Q_v}\vec{y} = \pm \|\vec{y}\|_2 \vec{e_1} \tag{7.26}$$

The solution of this basic task is:

$$\vec{v} = \vec{y} \pm \|\vec{y}\|_2 \vec{e_1} \tag{7.27}$$

In order to avoid loss of significance, $\vec{v}$ is selected as:

$$\vec{v} = \vec{y} + \text{sign}\,(y_1)\,\|\vec{y}\|_2 \vec{e_1}, \tag{7.28}$$

whereby y_1 is the first element of $\vec{y}$ and $\text{sign}\,(0) = 1$ [199]. The rules of the Householder are summarized as:

$$\begin{aligned} \alpha &= \text{sign}\,(y_1)\,\|\vec{y}\|_2 \\ \vec{v} &= \vec{y} + \alpha\vec{e_1} \\ \boldsymbol{Q_v}\vec{y} &= -\alpha\vec{e_1} \end{aligned} \tag{7.29}$$

7.2.5.3. Householder Principal

$$\boldsymbol{A} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \xrightarrow{Q_1} \boldsymbol{Q_1 A} = \begin{bmatrix} \circledast & \circledast & \circledast \\ 0 & \circledast & \circledast \\ 0 & \circledast & \circledast \\ 0 & \circledast & \circledast \end{bmatrix} \xrightarrow{Q_2} \boldsymbol{Q_2 Q_1 A} = \begin{bmatrix} * & * & * \\ 0 & \circledast & \circledast \\ 0 & 0 & \circledast \\ 0 & 0 & \circledast \end{bmatrix}$$

$$\xrightarrow{Q_3} \boldsymbol{Q_3 Q_2 Q_1 A} = \begin{bmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & \circledast \\ 0 & 0 & 0 \end{bmatrix}$$

The new computed matrix entries using the matrix $\boldsymbol{Q_v}$ are marked with $\circledast$.

7.2.5.4. Householder Example

For $\vec{y} = [2, 2, 1]^T$ is to find a $\vec{v} \in \mathbb{R}^3$, that applies:

$$\boldsymbol{Q_v}\vec{y} = \pm \|\vec{y}\|_2 \vec{e_1} = \pm 3 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},$$

we get $\alpha = 3$, $\vec{v} = \vec{y} + \alpha\vec{e_1}$, and so

$$\boldsymbol{Q_v} = \boldsymbol{I} - 2\frac{\vec{v}\vec{v}^T}{\vec{v}^T\vec{v}} = \frac{1}{15}\begin{bmatrix} -10 & -10 & 5 \\ -10 & 11 & -2 \\ -5 & -2 & 14 \end{bmatrix} \tag{7.30}$$

7.2.6. Singular Value Decomposition

The QR, as well as the SVD, use orthogonal transformations to reduce the least squares problem to a triangular and a diagonal system, respectively. Both methods provide a stable way of computing matrices, but QR methods may fail when the matrix $\boldsymbol{A}$ is nearly rank-deficient. Therefore, the SVD is recommended for rank-deficient as well as nearly rank-deficient matrices. An m by n matrix $\boldsymbol{A}$ can be decomposed as:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = (\textbf{orthogonal})(\textbf{diagonal})(\textbf{orthogonal}), \tag{7.31}$$

$$\text{with } \boldsymbol{A}\vec{v_i} = \sigma_i\vec{u_i}, \tag{7.32}$$

whereby the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are the left and right singular vectors, respectively. The diagonal elements of the matrix $\boldsymbol{\Sigma}$ are the singular values [209]:

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_r) \in \mathbb{R}^{mxn}, \tag{7.33}$$

whereby $r = \min(m, n)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$.

The SVD of the matrix $\boldsymbol{A}$ is very close to the eigenvalue-eigenvector $\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$ factorization of the positive definite matrix $\boldsymbol{A}^T\boldsymbol{A}$. The eigenvalues, as well as the eigenvectors, are in the diagonal matrix $\boldsymbol{\Lambda}$ and the orthogonal matrix $\boldsymbol{Q}$, respectively. An eigenvalue λ with the eigenvector $\vec{v}$ of the matrix $\boldsymbol{A}$ are a scalar and a nonzero vector that satisfy:

$$\boldsymbol{A}\vec{v} = \lambda\vec{v}. \tag{7.34}$$

$$(\boldsymbol{A} - \lambda\boldsymbol{I}) = \vec{0}. \tag{7.35}$$

In addition, the matrix $\boldsymbol{A}^T\boldsymbol{A}$ is semi-positive definite, since:

1. The matrix $\boldsymbol{A}^T\boldsymbol{A}$ is real and symmetric, therefore, the eigenvalues are real, and the eigenvectors form an orthonormal basis for $\mathbb{R}^n$.
2. The scalar: $\vec{x}^T(\boldsymbol{A}^T\boldsymbol{A})\vec{x} = (\boldsymbol{A}\vec{x})^T(\boldsymbol{A}\vec{x}) = \|\boldsymbol{A}\vec{x}\|_2 \geq 0$.

Therefore, the eigenvalues (λ_i) of the matrix $\boldsymbol{A}^T\boldsymbol{A}$ are positive ($\lambda_i \geq 0$), whereas:

$$\sigma_i = \sqrt{\lambda_i}. \tag{7.36}$$

Hence, the diagonal (but rectangular) matrix $\boldsymbol{\Sigma}$ has the eigenvalues from the matrix $\boldsymbol{A}\boldsymbol{A}^T$. Furthermore, the columns of the $(m \times m)$ matrix $\boldsymbol{U}$ are eigenvectors of the matrix $\boldsymbol{A}\boldsymbol{A}^T$, while the columns of the $(n \times n)$ matrix $\boldsymbol{V}$ are eigenvectors of the matrix $\boldsymbol{A}^T\boldsymbol{A}$ [209]. This can be demonstrated as follows:

$$\boldsymbol{A}\boldsymbol{A}^T = \left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\right)\left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\right)^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T\boldsymbol{U}^T, \tag{7.37}$$

whereas $\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T$ is the m by m eigenvalue matrix with $\sigma_1^2, \sigma_2^2, \cdots, \sigma_r^2$ on the diagonal. Similarly:

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{7.38}$$

where $\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$ is the n by n eigenvalue matrix with $\sigma_1^2, \sigma_2^2, \cdots, \sigma_r^2$ on the diagonal. Equation (7.38) implies:

$$\boldsymbol{A}^T\boldsymbol{A}\vec{v_i} = \sigma_i^2\vec{v_i} \tag{7.39}$$

Proof of: $\boldsymbol{A}\vec{v_i} = \sigma_i\vec{u_i}$. Multiplying (7.39) by the matrix $\boldsymbol{A}$:

$$\begin{aligned} \boldsymbol{A}\boldsymbol{A}^T\boldsymbol{A}\vec{v_i} &= \sigma_i^2\boldsymbol{A}\vec{v_i} \\ \left(\boldsymbol{A}\boldsymbol{A}^T\right)(\boldsymbol{A}\vec{v_i}) &= \sigma_i^2\left(\boldsymbol{A}\vec{v_i}\right), \end{aligned} \tag{7.40}$$

which means that $\boldsymbol{A}\vec{v_i}$ is an eigenvector of the matrix $\boldsymbol{A}\boldsymbol{A}^T$, whereby the length of the vector $\boldsymbol{A}\vec{v_i}$ can be calculated by multiplying (7.39) by the vector $\vec{v_i}^T$ as follows:

$$\begin{aligned} \vec{v_i}^T\left(\boldsymbol{A}^T\boldsymbol{A}\vec{v_i}\right) &= \vec{v_i}^T\left(\sigma_i^2\vec{v_i}\right) \\ (\boldsymbol{A}\vec{v_i})^T(\boldsymbol{A}\vec{v_i}) &= \sigma_i^2\vec{v_i}^T\vec{v_i} \\ \|\boldsymbol{A}\vec{v_i}\|_2 &= \sigma_i^2. \end{aligned} \tag{7.41}$$

Therefore, the unit eigenvector is:

$$\frac{\boldsymbol{A}\vec{v_i}}{\sigma_i} = \vec{u_i}, \tag{7.42}$$

and it applies:

$$\boldsymbol{AV} = \boldsymbol{\Sigma U}. \tag{7.43}$$

7.2.6.1. Computation of the Singular Value Decomposition

The calculation of the SVD is costlier than the computation of the QR decomposition either by the Gevins or the Householder algorithm [205]. Computing the SVD by finding the eigenvalues of the symmetric matrix $\boldsymbol{A}^T\boldsymbol{A}$, as seen in the previous Section 7.2.6, is not a numerically effective method due to round-off errors in the calculation of the matrix $\boldsymbol{A}^T\boldsymbol{A}$ [205]. Therefore, there are various algorithms that implement the SVD in a numerically effective way.

The Golub–Kahan–Reinsch (GKR) algorithm, which works in two steps, is a widely used method for computing the SVD [205]. In the first step, the matrix $\boldsymbol{A}$ is reduced to a bidiagonal matrix $\boldsymbol{B}$, while in the second step, the matrix $\boldsymbol{B}$ is further reduced to a diagonal matrix of singular values. Other algorithms are the Chan SVD, Demmel–Kahan Zero-Shift QR, the Differential Quotient Difference (DQD) algorithm of Fernando and Parlett [205].

7.2.7. Summary and Comparison of Decomposition Algorithms

There are four important matrix decompositions: Cholesky, LU, QR, and SVD. The Cholesky decomposition is used to solve the normal equation, which can become unstable due to the rounding errors corresponding to $[\kappa(\boldsymbol{A})]^2$ instead of $\kappa(\boldsymbol{A})$.

The LU factorization can be achieved by using the Gaussian Elimination (GE). GE is efficient, since it requires only $\frac{n^3}{3}$ flops, but it is unstable for arbitrary matrices. The use of the GE without pivoting is not recommended in practice except for column diagonally dominant or symmetric positive definite matrices [205]. Furthermore, the LU decomposition is limited to square matrices.

The QR factorization can be applied to every matrix, which means the matrix can be rectangular. Furthermore, pivoting is not required to admit a stable QR decomposition of a matrix. The QR factorization is unique if the matrix is not singular. The QR decomposition can be calculated by using the Classical Gram–Schmidt (CGS), Modified Gram–Schmidt (MGS), the Givens rotation, or the Householder algorithm. The CGS is numerically unstable, while the MGS has a better stability property, but it is not as stable as the Gevins or the Householder methods [205]. The Householder is more efficient than the Givens, since they need $2n^2(m - n/3)$ and $3n^3(m - n/3)$ flops, respectively. Both algorithms have a guaranteed stability but fail if the matrix is singular or nearly rank-deficient.

The SVD is more expensive than the Givens or the Householder algorithms, but it is numerically stable and can handle the rank deficiency. The GKR method needs about $4m^2n + 8mn^2 + 9n^3$ flops to compute the SVD of an m by n matrix.

A comparison of the algorithms building the matrix decompositions is summarized in Table 7.1.

Table 7.1.: *Comparison of decomposition algorithms for an* $(m \times n)$ *matrix* $\boldsymbol{A}$

Algorithms	Decomposition	Pivoting	Stability	Cost	Remark
Cholesky for normal equation	$\mathbf{AA}^T = \mathbf{LDL}^T$	no	Can be unstable	$n^2m + \frac{n^3}{3}$	Instability due rounding errors
Gauss elimination without pivoting	$\mathbf{A}=\mathbf{LU}$	no	Unstable	$\frac{2n^3}{3}$	Limited to quadratic matrices
Gauss elimination with pivoting	$\mathbf{A}=\mathbf{LU}$	yes	Stable	$\frac{2n^3}{3}$	Limited to quadratic matrices
Classical Gram-Schmidt (CGM)	$\mathbf{A}=\mathbf{QR}$	no	Unstable	$2mn^2$	Possible severe loss of orthogonality
Modified Gram-Schmidt	$\mathbf{A}=\mathbf{QR}$	no	better than GM	$2mn^2$	Not as stable as Givens or Householder methods
Givens	$\mathbf{A}=\mathbf{QR}$	no	Stable	$3n^3(m - \frac{n}{3})$	Cannot handle rank deficiency
Householder	$\mathbf{A}=\mathbf{QR}$	no	Stable	$2n^2(m - \frac{n}{3})$	Cannot handle rank deficiency
Golub-Kahan-Reinsch	$\mathbf{A}=\boldsymbol{U\Sigma V}^T$	no	Stable	$4m^2n+$ $8mn^2+$ $9n^3$	The algorithm is based on SVD

7.2.8. Memory Requirements

Memory requirements are considered only for the Givens, Householder, and GKR SVD methods, since they are stable as well as enable the solution of overdetermined linear systems. The Givens matrices can be calculated implicitly; in other words, it is not necessary to build them. Furthermore, the matrix $\boldsymbol{R}$ can be constructed by overwriting the matrix $\boldsymbol{A}$. Therefore, the memory necessary for the Givens method is approximately equal to $m \times n$, which is needed to calculate the matrix $\boldsymbol{Q}$. As is the case with the Givens method, the construction of Householder matrices is not necessary, and the required memory space is also approximately equal to $m \times n$. In addition to the matrices $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$, the GKR SVD needs three $n \times n$ matrices. These square matrices are used for the Householder reduction method as well as the Golub–Kahan SVD step [210]. Therefore, the memory needed for the GKR SVD is approximately equal to: $4(n \times n) + 2(m \times m)$.

7.3. Position Optimization Algorithms

In the previous Section 7.2, we showed how to find an approximate position by solving (7.2). In this section, we use optimization algorithms to refine the accuracy of the approximate position. The optimization can be achieved by using non-linear least squares methods. The NLS method optimizes the position (solution) computed by (7.2). These methods are iterative, whereby each iteration step needs a solution of the corresponding linear least squares problem. The NLS problem is a special case of the general optimization problem in $\mathbb{R}^n$ [197]. The NLS approach is based on the minimization of the squares of the errors of the measured distances or the strength of the magnetic field, see Section 5.2.2.3 and 6.6.2, respectively. Thus, the following function must be minimized:

$$F(x, y, z) = \sum_{i=1}^{n} f_i^2(x, y, z) \tag{7.44}$$

Whereby, $f_i(x, y, z)$ is the error function. Minimizing the sum of the square errors is a common problem in applied mathematics, which can be solved, for instance, with the Gauss–Newton or the Levenberg–Marquardt algorithm [169].

7.3.1. Non-Linear Least Squares Problems

The equations (5.1) as well as (6.7) are an over-determined system of equations, which should be conveyed to a NLS problem. The general NLS problem that should be solved for the UWB-ILS and for MILPS is:

$$\vec{x}^* \in \mathbb{R}^3 \ , \ \|\vec{f}(\vec{x}^*)\|_2 = \min_{\vec{x} \in \mathbb{R}^3} \|\vec{f}(\vec{x})\|_2 \, . \tag{7.45}$$

The NLS problem of the UWB-ILS as well as of the MILPS will be presented in Sections 7.3.1.1 and 7.3.1.2, respectively.

7.3.1.1. UWB-ILS: Non-Linear Least Squares Problem

The algebraic multilateration method does not always provide a good estimation due to the measurement uncertainties [168]. Therefore, we define the error function $f_{i,1}(x, y, z)$ of the UWB-ILS as follows:

$$f_{i,1}(x, y, z) = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} - d_i \qquad i = 1, ..., n. \tag{7.46}$$

7.3.1.2. MILPS: Non-Linear Least Squares Problem

The error function of the MILPS is:

$$f_{i,2}(x,y,z) = \frac{k}{r_i^3}\sqrt{1+3\sin^2\theta_i} - B_i = 0 \quad i = 1,2,...,m \tag{7.47}$$

Starting from an approximation solution $\vec{x}^{(0)}$, the iterative approach creates a series of $\{\vec{x}^{(i+1)}\}_{i\geq 0}$, which converge to the solution of the problem. The start vector $\vec{x}^{(0)}$ is calculated by using the multilateration algorithm based on the SVD. Since the SVD can deliver a good solution in the sense of the least squares method, the start vector $\vec{x}^{(0)}$ can be the searched solution. If the start position $\vec{x}^{(0)}$ should be optimized, the optimization algorithm is used beginning from the start position $\vec{x}^{(0)}$.

7.3.2. Jacobi-Matrices

As mentioned in Section 5.2.2.3 and 6.6.2, the Jacobian matrices of UWB-ILS as well as for MILPS are derived in a convenient form for resource-constrained devices. The Gauss–Newton as well as the Levenberg–Marquardt method require the first derivatives; therefore, we define the following Jacobian matrix:

$$\boldsymbol{J_f} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial y} & \frac{\partial f_i}{\partial z} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x} & \frac{\partial f_n}{\partial y} & \frac{\partial f_n}{\partial z} \end{pmatrix}, \tag{7.48}$$

whereby the error function vector $\vec{f}$ is:

$$\vec{f} = \begin{pmatrix} f_1, & f_2, & f_3, & \ldots, & f_n \end{pmatrix}^T. \tag{7.49}$$

The Jacobian matrices of the UWB-ILS as well as of MILPS are defined in (5.12),(5.13) and (6.11),(6.12),(6.13) respectively.

7.3.3. Gauss–Newton Algorithm

The Gauss–Newton method is a numerical approach that can be used for solving NLS problems. The GNM proceeds by using the error function f_i and the Jacobian matrix $\boldsymbol{J_f}$ as follows:

1. Choose a convenient start position $\vec{x}^{(0)}$, which is computed using the multilateration algorithm based on the SVD defined in Section 7.2.6.

2. Consider the following iteration scheme:

$$\vec{x}^{(i+1)} = \vec{x}^{(i)} + \vec{s}^{(i)} \quad , i = 0, 1, 2, ..., \tag{7.50}$$

whereby the vector $\vec{s}^{(i)}$ is the solution of the following linear system:

$$\left(\boldsymbol{J}_{\boldsymbol{f}}^T \boldsymbol{J}_{\boldsymbol{f}}\right) \vec{s}^{(i)} = -\boldsymbol{J}_{\boldsymbol{f}}^T \vec{f}(\vec{x}^{(i)}). \tag{7.51}$$

The position $\vec{x}^{(i)}$ is accepted as an approximate solution, if the following condition is satisfied:

$$\|\vec{s}^{(i)}\|_2 \leq \varepsilon_x \cdot (1 + \|\vec{x}^{(i)}\|_2),$$

whereas ε_x is the accuracy bound, which can be selected by the user. A further condition is required: $i > i_{max}$ to avoid an infinite loop. The maximal number of iterations i_{max} is also selected by the user.

The GNM can achieve good performance, but only if the matrix $\boldsymbol{J}_{\boldsymbol{f}}^T \boldsymbol{J}_{\boldsymbol{f}}$ will not be singular, that means if $\boldsymbol{J}_{\boldsymbol{f}}^T \boldsymbol{J}_{\boldsymbol{f}}$ has a full rank. In some cases, the matrix $\boldsymbol{J}_{\boldsymbol{f}}^T \boldsymbol{J}_{\boldsymbol{f}}$ tends to be singular after only a few iterations, thus the method diverges. In our case, Equation (7.51) is solvable, if and only if the (3×3) matrix $\boldsymbol{J}_{\boldsymbol{f}}^T \boldsymbol{J}_{\boldsymbol{f}}$ is invertible (has a full rank). Hence, the convergence of the GNM is not always guaranteed, it makes this method unsuitable for solving the NLS problem (7.45) for a practical and accurate method of computing a position.

An alternative approach is provided by the Levenberg–Marquardt method, which will be discussed in the next Section 7.3.4.

7.3.4. Levenberg–Marquardt Algorithms

The Levenberg–Marquardt algorithm belongs to the class of numerical optimization approaches for solving NLS problems using the least squares method. The LVM algorithm combines the GNM with a regularization parameter $\mu > 0$ to guarantee the robustness of the method. The scalar $\mu > 0$ is also called the damping parameter [204]. Two variants of the LVM algorithms will be discussed in Sections 7.3.4.1 and 7.3.4.2.

7.3.4.1. Dahmen-Reusken Levenberg–Marquardt Algorithm

The NLS problem [199] to be solved is

$$\arg\min_{\vec{x}} \left\| \begin{pmatrix} \boldsymbol{J}_{\boldsymbol{f}}(\vec{x}) \\ \mu \boldsymbol{I} \end{pmatrix} \vec{s} + \begin{pmatrix} \vec{f}(\vec{x}) \\ 0 \end{pmatrix} \right\|_2^2, \tag{7.52}$$

which can be transformed into the following equation:

$$(\boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x}), \mu \boldsymbol{I}) \begin{pmatrix} \boldsymbol{J}_{\boldsymbol{f}}(\vec{x}) \\ \mu \boldsymbol{I} \end{pmatrix} \vec{s} = -(\boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x}), \mu \boldsymbol{I}) \begin{pmatrix} f(\vec{x}) \\ 0 \end{pmatrix}, \tag{7.53}$$

it follows:

$$(\boldsymbol{J}_{\boldsymbol{f}}^{T}(\vec{x})\boldsymbol{J}_{\boldsymbol{f}}(\vec{x}) + \mu^2\boldsymbol{I})\vec{s} = -\boldsymbol{J}_{\boldsymbol{f}}^{T}(\vec{x})\vec{f}(\vec{x}) \tag{7.54}$$

It has the advantage over the GNM that the matrix in the left side of (7.54) is no longer singular or nearly singular. This is achieved by the regularization of the matrix $\boldsymbol{J}_{\boldsymbol{f}}^{\mathrm{T}}\boldsymbol{J}_{\boldsymbol{f}}$ with the factor $\mu^2\boldsymbol{I}$, whereby $\boldsymbol{I}$ is a (3×3) identity matrix. Hence, the matrix $\binom{\boldsymbol{J}_{\boldsymbol{f}}(\vec{x})}{\mu\boldsymbol{I}}$ has always a full rank and, therefore, a unique solution. Equation (7.54) has a unique solution, since the matrix $(\boldsymbol{J}_{\boldsymbol{f}}^{T}(\vec{x})\boldsymbol{J}_{\boldsymbol{f}}(\vec{x}) + \mu^2\boldsymbol{I})$ is positive definite [199].

It applies [199]:

$$\|\vec{s}^{(i)}\|_2 \leq \frac{\|\vec{f}(\vec{x}^{(i)})\|_2}{\mu}\,. \tag{7.55}$$

As can be seen from Equation (7.55), the damping parameter ($\mu > 0$) controls the correction vector $\vec{s}^{(i)}$. Small or large corrections can be achieved by a suitable choice of the value of μ in each iteration step. The parameter μ is denoted as $\mu^{(i)}$ to clarify its dependence on the iteration steps.

The approximate solution of the position $\vec{x}$ is calculated in each iteration step as in (7.50), where $\vec{s}^{(i)}$ is the solution of the following linear system:

$$(\boldsymbol{J}_{\boldsymbol{f}}^{T}(\vec{x}^{(i)})\boldsymbol{J}_{\boldsymbol{f}}(\vec{x}^{(i)}) + (\mu^{(i)})^2\boldsymbol{I})\vec{s}^{(i)} = -\boldsymbol{J}_{\boldsymbol{f}}^{T}(\vec{x}^{(i)})\vec{f}(\vec{x}^{(i)}) \tag{7.56}$$

We solve (7.56) by using the Householder instead of the SVD algorithm to save memory stack and computing time on resource-constrained devices. This is possible due to the robustness of the LVM method.

To guarantee the convergence of the LVM algorithm, the value of $\mu^{(i)}$ should not be selected too large or too small. For example, a large value of $\mu^{(i)}$ leads to a small correction, hence the LVM algorithm will slowly converge, if $\vec{x}^{(i)}$ is still far from the solution [199]. For this reason, a strategy for controlling the damping parameter μ is introduced. In this way, the actual value of $\mu^{(i)}$ is augmented, reduced, or maintained in each iteration step by using the parameter ρ_μ defined as:

$$\rho_\mu = \frac{\|\vec{f}(\vec{x}^{(i)})\|_2^2 - \|\vec{f}(\vec{x}^{(i)} + \vec{s}^{(i)})\|_2^2}{\|\vec{f}(\vec{x}^{(i)})\|_2^2 - \|\vec{f}(\vec{x}^{(i)}) + \boldsymbol{J}_{\boldsymbol{f}}(\vec{x}^{(i)})\vec{s}^{(i)})\|_2^2} =: \frac{\Delta R(\vec{x}^{(i)}, \vec{s}^{(i)})}{\Delta\tilde{R}(\vec{x}^{(i)}, \vec{s}^{(i)})}\,, \tag{7.57}$$

whereby $\Delta\tilde{R}(\vec{x}^{(i)}, \vec{s}^{(i)}) \geq 0$ as well as $\Delta R(\vec{x}^{(i)}, \vec{s}^{(i)}) > 0$ [199].

The strategy used for controlling the parameter μ is defined as:

$$\begin{cases} \rho_\mu \leq \beta_0 & \vec{s}^{(i)} \text{ is not accepted. Set } \mu^{(i)} = 2\mu^{(i)} \text{ and compute again } \vec{s}^{(i)} \\ \beta_0 < \rho_\mu < \beta_1 & \vec{s}^{(i)} \text{ is accepted. Use } \mu^{(i)} \text{ for } \vec{s}^{(i+1)} \\ \rho_\mu \geq \beta_1 & \vec{s}^{(i)} \text{ is accepted. Use } \dfrac{\mu^{(i)}}{2} \text{ for } \vec{s}^{(i+1)} \end{cases} \tag{7.58}$$

whereby $0 < \beta_0 < \beta_1 < 1$ (e.g., $\beta_0 = 0.2$, $\beta_1 = 0.8$).

The experience has shown that the selection of the initial value $\mu^{(0)}$ impacts the accuracy as well as the cost of the LVM algorithm (*cf.* Sections 7.5.4.3 and 7.7.3). In this context, it is difficult to find a well-founded strategy in the literature for the selection of the initial value $\mu^{(0)}$. A start value of $\mu^{(0)} = 0.01$ is proposed in [211]. Another strategy is suggested in [212, 213], whereby $\mu^{(0)}$ is equal to 1 or 10, if the initial guess $\vec{x}^{(0)}$ is believed to be a bad approximation. In contrast, $\mu^{(0)}$ is equal to 0.001 or 0.010, if the initial guess $\vec{x}^{(0)}$ is believed to be a good approximation. In this work, the strategy suggested by [204] is used. In this strategy, the initial value $\mu^{(0)}$ is calculated based on the matrix $\boldsymbol{A}^{(0)}$:

$$\boldsymbol{A}^{(0)} = \boldsymbol{J}_{\boldsymbol{f}}^{\mathrm{T}(0)} \boldsymbol{J}_{\boldsymbol{f}}^{(0)} \tag{7.59}$$

as follows:

$$\mu^{(0)} = \tau \cdot max_i\{a_{ii}^{(0)}\}, \tag{7.60}$$

where $a_{ii}^{(0)}$ are the diagonal elements of the matrix $\boldsymbol{A}^{(0)}$ and τ is chosen by the user. As a rule of thumb, a small value of τ should be chosen (e.g., $\tau = 10^{-6}$), if the initial guess $\vec{x}^{(0)}$ is believed to be a good approximation; otherwise, $\tau = 10^{-3}$ or $\tau = 1$ can be used.

The DR-LVM algorithm terminates such as the GNM, if the following conditions are satisfied:

1. $\|\vec{s}^{(i)}\|_2 \leq \varepsilon_x \cdot (1 + \|\vec{x}^{(i)}\|_2)$
2. $i \geq i_{max}$,

whereby, ε_x is the accuracy bound that is selected from the user. In summary, the algorithm can be described by the pseudo-code of Algorithm 1.

As can be seen from the Algorithm 1, the DR-LVM algorithm is recursive. Therefore, the recursive DR-LVM method is changed in an iterative procedure, in order to port it to limited stack memory devices such as microcontrollers. The iterative approach has an additional *while*-loop compared to the recursive method, as demonstrated in Algorithm 2.

7.3.4.2. Madsen Levenberg–Marquardt Algorithm

Another variant of the LVM algorithm is proposed by Madsen *et al.* [204], whereby the NLS problem to be solved is

$$\arg\min_{\vec{x}} \left\| \begin{pmatrix} \boldsymbol{J_f}(\vec{x}) \\ \sqrt{\mu}\boldsymbol{I} \end{pmatrix} \vec{s} + \begin{pmatrix} \vec{f}(\vec{x}) \\ 0 \end{pmatrix} \right\|_2^2. \tag{7.61}$$

The search direction $\vec{s}^{(i)}$ is calculated recursively ($\vec{x}^{(i+1)} = \vec{x}^{(i)} + \vec{s}^{(i)}$) according to the following formula:

$$(\boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x}^{(i)})\boldsymbol{J_f}(\vec{x}^{(i)}) + \mu^{(i)}\boldsymbol{I})\vec{s}^{(i)} = -\boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x}^{(i)})\vec{f}(\vec{x}^{(i)}). \tag{7.62}$$

Algorithm 1 Dahmen-Reusken LVM algorithm (recursive)

function DR_LVM_ALG($\varepsilon_x, \beta_0, \beta_1, \vec{x}^{(0)}, \tau, i_{max}$)
  $i = 0$; $\vec{x} = \vec{x}^{(0)}$; $\boldsymbol{B} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\boldsymbol{J}_{\boldsymbol{f}}(\vec{x})$; $\vec{H} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\vec{f}(\vec{x})$;
  $\mu^{(0)} = \tau \cdot \max_i \{b_{ii}(\vec{x})\}$; $\mu = \mu^{(0)}$;
  Solve $(\boldsymbol{B} + \mu^2\boldsymbol{I})\vec{s} = -\vec{H}$ for $\vec{s}$;
  while $((\|\vec{s}\|_2 > \varepsilon_x(1 + \|\vec{x}\|_2)$ **and** $(i < i_{max}))$ **do**
    $[\vec{s}, \rho_\mu]$ = CORRECTION$(\vec{x}, \mu, \beta_0, \beta_1)$;
    $\vec{x} = \vec{x} + \vec{s}$;
    $i = i + 1$;
  end while
end function

function CORRECTION$(\vec{x}, \mu, \beta_0, \beta_1)$
  $\boldsymbol{B} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\boldsymbol{J}_{\boldsymbol{f}}(\vec{x})$; $\vec{H} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\vec{f}(\vec{x})$;
  Solve $(\boldsymbol{B} + \mu^2\boldsymbol{I})\vec{s} = -\vec{H}$ for $\vec{s}$;
  $\rho_\mu = \frac{\|\vec{f}(\vec{x})\|_2^2 - \|\vec{f}(\vec{x}+\vec{s})\|_2^2}{\|\vec{f}(\vec{x})\|_2^2 - \|\vec{f}(\vec{x}) + \boldsymbol{J}_{\boldsymbol{f}}(\vec{x})\vec{s})\|_2^2}$;
  if $(\rho_\mu \leq \beta_0)$ **then**
    $[\vec{s}, \rho_\mu]$ = CORRECTION$(\vec{x}, 2\mu, \beta_0, \beta_1)$;
  else if $(\rho_\mu \geq \beta_1)$ **then**
    $\mu = \frac{\mu}{2}$
  end if
end function

As previously mentioned, we compute (7.62) by using the Householder algorithm to save memory stack as well as computing time.

The initial value of the damping parameter $\mu^{(0)}$ is calculated according to Equation (7.60). Furthermore, the value of $\mu^{(i+1)}$ for the next iteration depends on the ratio ϱ, which is defined as:

$$\varrho = \frac{\frac{1}{2}(\|\vec{f}(\vec{x}^{(i)})\|_2^2 - \|\vec{f}(\vec{x}^{(i+1)})\|_2^2)}{\frac{1}{2}(\vec{s}^{(i)})^T(\mu^{(i)}\vec{s}^{(i)} - \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x}^{(i)})\vec{f}(\vec{x}^{(i)}))}, \tag{7.63}$$

The algorithm terminates if the change in $\vec{x}^{(i)}$ is very small:

$$\|\vec{x}^{(i+1)} - \vec{x}^{(i)}\|_2 \leq \varepsilon_1(\|\vec{x}^{(i)}\|_2 + \varepsilon_2), \tag{7.64}$$

or, if the following condition is fulfilled:

$$\|\boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\vec{f}(\vec{x})\|_2 \geq \varepsilon_1. \tag{7.65}$$

An additional termination criterion is required to avoid an endless loop is: $i \geq i_{max}$. The accuracy bounds ε_1, ε_2, and i_{max} are selected by the user. In summary, the algorithm can be described by the pseudo-code in Algorithms 3 and 4.

Algorithm 2 Dahmen-Reusken LVM algorithm (iterative)

function DR_LVM_ALG($\varepsilon_x, \beta_0, \beta_1, \vec{x}^{(0)}, \tau, i_{max}$)
  $i = 0$; $\vec{x} = \vec{x}^{(0)}$; $\boldsymbol{B} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\boldsymbol{J}_{\boldsymbol{f}}(\vec{x})$; $\vec{H} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\vec{f}(\vec{x})$;
  $\mu^{(0)} = \tau \cdot \max_i \{b_{ii}(\vec{x})\}$; $\mu = \mu^{(0)}$;
  Solve $(\boldsymbol{B} + \mu^2\boldsymbol{I})\vec{s} = -\vec{H}$ for $\vec{s}$;
  while (($\|\vec{s}\|_2 > \varepsilon_x(1 + \|\vec{x}\|_2)$ **and** $(i < i_{max})$) **do**
    $[\vec{s}, \rho_\mu]$ = CORRECTION($\vec{x}, \mu, \beta_0, \beta_1$);
    while (true) **do**
      if $(\rho_\mu \leq \beta_0)$ **then**
        $\mu = 2\mu$
        $[\vec{s}, \rho_\mu]$ = CORRECTION($\vec{x}, \mu, \beta_0, \beta_1$);
      else if $(\rho_\mu \geq \beta_1)$ **then**
        $\mu = \frac{\mu}{2}$
        break;
      else
        break;
      end if
    end while
    $\vec{x} = \vec{x} + \vec{s}$;
    $i = i + 1$;
  end while
end function

function CORRECTION($\vec{x}, \mu, \beta_0, \beta_1$)
  $\boldsymbol{B} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\boldsymbol{J}_{\boldsymbol{f}}(\vec{x})$; $\vec{H} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\vec{f}(\vec{x})$;
  Solve $(\boldsymbol{B} + \mu^2\boldsymbol{I})\vec{s} = -\vec{H}$ for $\vec{s}$;
  $\rho_\mu = \frac{\|\vec{f}(\vec{x})\|_2^2 - \|\vec{f}(\vec{x}+\vec{s})\|_2^2}{\|\vec{f}(\vec{x})\|_2^2 - \|\vec{f}(\vec{x}) + \boldsymbol{J}_{\boldsymbol{f}}(\vec{x})\vec{s})\|_2^2}$;
end function

Algorithm 3 Madsen LVM algorithm [204], Part 1

function MADS_LVM_ALG($\varepsilon_1, \varepsilon_2, \vec{x}^{(0)}, \tau, i_{max}$)
  $i = 0$; $\vec{x} = \vec{x}^{(0)}$; $\boldsymbol{B} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\boldsymbol{J}_{\boldsymbol{f}}(\vec{x})$; $\vec{H} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\vec{f}(\vec{x})$;
  $\mu^{(0)} = \tau \cdot \max_i \{b_{ii}(\vec{x})\}$; $\mu = \mu^{(0)}$;
  while (($|\vec{H}\|_2 > \varepsilon_1$) **and** $(i < i_{max})$) **do**
    $i = i + 1$; $\mu = \mu^{(0)}$;
    Solve $(\boldsymbol{B} + \mu\boldsymbol{I})\vec{s} = -\vec{H}$ for $\vec{s}$;
    if $(\|\vec{s}\|_2 \leq \varepsilon_2(\|\vec{x}^{(i)}\|_2 + \varepsilon_2))$ **then**
      break;
    else
      $\vec{x}_{new} = \vec{x} + \vec{s}$; $\varrho = \frac{\frac{1}{2}(\|\vec{f}(\vec{x})\|_2^2 - \|\vec{f}(\vec{x}_{new})\|_2^2)}{\frac{1}{2}\vec{s}^T(\mu\vec{s}\boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\vec{f}(\vec{x}))}$
      if $(\varrho > 0)$ **then**
        $\vec{x} = \vec{x}_{new}$; $\boldsymbol{B} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\boldsymbol{J}_{\boldsymbol{f}}(\vec{x})$; $\vec{H} = \boldsymbol{J}_{\boldsymbol{f}}^T(\vec{x})\vec{f}(\vec{x})$;
        $\mu = \mu \cdot \max\{\frac{1}{3}, 1 - (2\varrho - 1)^3\}$; $v = 2$;
      else

Algorithm 4 Madsen LVM algorithm, Part 2

```
15:                 μ = μ · v;  v = 2 · v;
16:             end if
17:         end if
18:     end while
19: end function
```

7.3.5. Multipath Distance Detection and Mitigation and Position Optimization Algorithm

Although UWB technology allows an accurate range estimation, its accuracy is altered by the problems of NLoS such as blockage situations or multipath [214]. Multipath fading is caused by wave reflection or diffraction from objects such as walls, ceilings, or moving persons inside a closed room. In the worst case scenario, the multipath signal can be interpreted as a direct path signal leading to false distance measurement. Prieto *et al.* proposed the so-called Robust Position Estimation in Ultrasound Systems (RoPEUS) algorithm to reduce multipath effects in an ultrasonic positioning system [215]. We used a reduced version of the RoPEUS algorithm, since UWB signals are less sensitive to multipath effects than ultrasonic signals. This algorithm uses the Least Trimmed Squares (LTS) estimator, which combines the robust regression with outlier detection method [216]. The main differences to our approaches are: We compute the position by using the multilateration method based on SVD, whereby the LVM algorithm is only used if an unfavorable geometric configuration occurs. In other words, the LVM method is invoked, if a PDOP threshold value is exceeded. Furthermore, our algorithm is optimized for devices with limited stack memory.

The flowchart in Figure 7.4 illustrates our algorithm. The observed distance measurements to n anchors are split into all possible sub-experiments of distance measurements to k anchors ($k < n$). The total number of possible experiments is equal to $\binom{n}{k}$.

In the first step, we select sub-experiments of distance measurements to k reference points, then we compute the estimated position using the multilateration algorithm for all k combinations as well as the corresponding n residuals r_i, as follows:

$$r_i = R_i - d_i, \quad i = 0, 1, \ldots, n, \tag{7.66}$$

where, R_i are the calculated distances from the MS to the RSs and d_i are the measured distances. Finally, the k smallest squared residuals are summed: from $\binom{n}{k}$ possible r_i, a smallest one is saved in the variable r_{min}. The position associated with r_{min} is the solution of the algorithm. By this procedure, up to n-k corrupted measurements could be discarded.

In the second step, we check the geometrical configuration of the chosen reference points: If the calculated PDOP value is bigger than a given threshold (see Section 7.5.4.2), we refine the solution by using the LVM algorithm, whereby the position computed from the Multipath Distance Detection and Mitigation (MDDM) algorithm is used as a start solution.

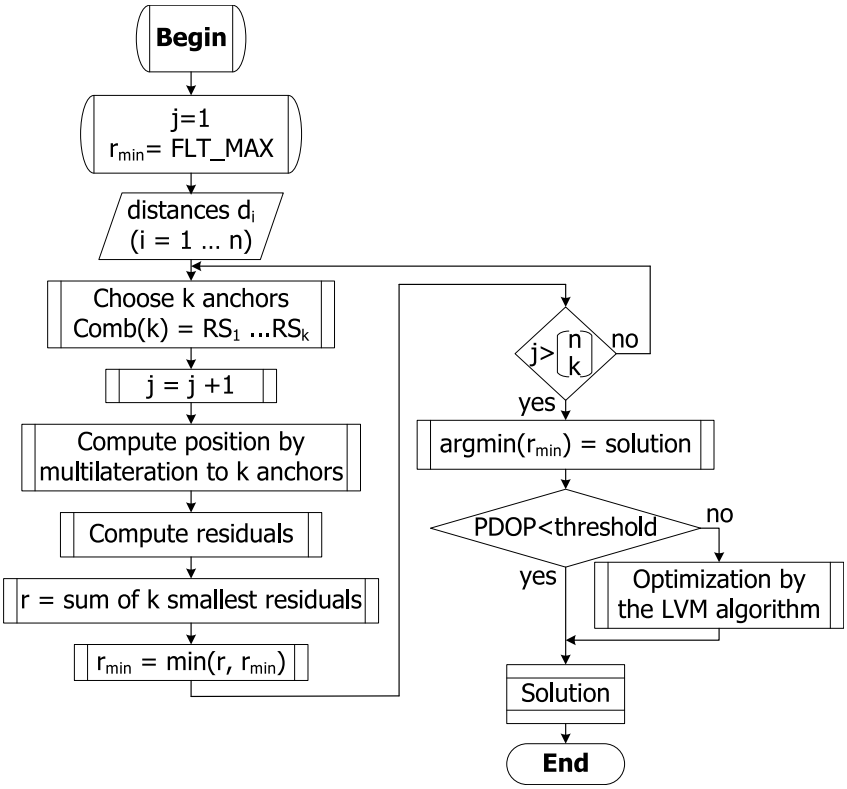


Figure 7.4.: *Flowchart of the multipath distance detection and mitigation, and position optimization algorithm*

7.4. Position Dilution of Precision and Empirical Cumulative Distribution Function

We perform test cases based on PDOP values to develop a method, which decides whether the LVM algorithm should be used or not. We briefly describe the PDOP as well as the Empirical Cumulative Distribution Function (ECDF) to evaluate the algorithms tested.

7.4.1. Position Dilution of Precision

The quality of the position is given by PDOP. The lower its value, the higher a positioning accuracy can be achieved. PDOP can be calculated as follows [217, 218]:

$$\boldsymbol{M} = \begin{bmatrix} \frac{x_1 - x}{R_1} & \frac{y_1 - y}{R_1} & \frac{z_1 - z}{R_1} & -1 \\ \frac{x_2 - x}{R_2} & \frac{y_2 - y}{R_2} & \frac{z_2 - z}{R_2} & -1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{x_m - x}{R_m} & \frac{y_m - y}{R_m} & \frac{z_m - z}{R_m} & -1 \end{bmatrix}, \tag{7.67}$$

where $R_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}$. Furthermore, (x, y, z) and (x_i, y_i, z_i) are the positions of the searched MSs and the reference stations, respectively.

$$\boldsymbol{Q_p} = (\boldsymbol{M}^T \boldsymbol{M})^{-1} = \begin{bmatrix} q_x^2 & q_{xy}^2 & q_{xz}^2 & q_{xt}^2 \\ q_{xy}^2 & q_y^2 & q_{yz}^2 & q_{yt}^2 \\ q_{xz}^2 & q_{yz}^2 & q_z^2 & q_{zt}^2 \\ q_{xt}^2 & q_{yt}^2 & q_{zt}^2 & q_t^2 \end{bmatrix} \tag{7.68}$$

The value of PDOP is calculated by using the first four diagonal elements of the matrix $\boldsymbol{Q_p}$:

$$PDOP = \sqrt{q_x^2 + q_y^2 + q_z^2}. \tag{7.69}$$

In this work, the matrix $\boldsymbol{Q_p}$ is computed by using the pseudo-inverse technique based on the SVD to avoid the problem of rank deficiency.

7.4.2. Empirical Cumulative Distribution Function

The results of the test cases are illustrated and evaluated by using the ECDF. The ECDF gives a global view of the errors in the x-, y-, and z-directions. The errors are calculated based on the absolute error, which is the difference between the searched and the calculated position: $|\text{approximate position} - P_i|$, whereby P_i is the exact position. The ECDF $F(e)$ is defined as:

$$F(e) = \begin{cases} 0 & e < e_1 \\ F_i = F(e_i) & e_i \leq e < e_{i+1},\ i = 1, 2, ...k-1, \\ 1 & e \geq e_k \end{cases} \tag{7.70}$$

where $0 \leq F(e) \leq 1$ and F_i is the relative cumulative frequency:

$$F_i = \sum_{j=1}^{i} f_i = \sum_{j=1}^{i} f(e_j) = f(e_1) + f(e_2) + ... + f(e_i), \tag{7.71}$$

whereby, f_i is the relative frequency:

$$f_i = \frac{h_i}{U} = \frac{\text{absolute frequency}}{\text{sample size}}, \tag{7.72}$$

where U is the number of the calculated absolute errors e_i, and h_i specifies how often the absolute error e_i occurs in the sample U_s.

7.5. Evaluation of UWB-ILS in a Simulated Environment

In this section, various test cases are performed to find out, when the execution of the LVM algorithm is necessary on the basis of the PDOP values. These tests are accomplished by sampling a closed room, whereby each point corresponds to two positions: a nominal and an approximate position. The approximate positions are calculated by using randomly noisy distances. First, we present a basic sampling configuration, then we introduce test cases with eight, four, and finally with an increasing number of references stations (from five up to eight reference stations). In addition, a parameter analysis is achieved to assess the parameter combination enabling the best performance. The results of each test scenario are evaluated and graphically presented using the ECDFs. The examination of all possible RS configurations to determine a generally applicable threshold of PDOP is not possible and is not the focus of this thesis. Usually, the position of the RSs can be freely selected to ensure good coverage [11]. Therefore, we propose a way how to determine the PDOP threshold in the next subsections.

7.5.1. Environments and Parameters Used

The LVM algorithms will be analyzed and compared in simulated and real environments. Furthermore, the GNM is compared with the LVM algorithm in a real environment. The simulated tests are accomplished and evaluated in MATLAB. The real tests are achieved in the STM32F4 microcontroller as well as in Raspberry Pi 3 but they are evaluated in MATLAB. The unoptimized position $\vec{x}^{(0)}$ (start position) is calculated by using the multilateration algorithm defined in Section 7.1. The multilateration approach is based on the SVD method [14]. The precision of the optimization algorithms is denoted with ε; then one has: $\varepsilon = \varepsilon_x = \varepsilon_1 = \varepsilon_2$. The GNM is implemented with ε equal to 10^{-5}, while the DR-LVM and M-LVM algorithms are performed and compared with ε-values from 10^{-6} up to 10^{-1}. The maximal iteration numbers (i_{max}) are set to 100 for all optimization algorithms. The DR-LVM algorithm is performed with: $\beta_0 = 0.2$ and $\beta_1 = 0.8$; these parameters are not required for the M-LVM method. Both variants of LVM algorithms are implemented and tested with the following τ-values: 1, 10^{-3} and 10^{-6}. We set all the accuracy bounds equal ($\varepsilon_x = \varepsilon_1 = \varepsilon_2 = \varepsilon$) as well as the maximal iteration equal to 100 to compare the algorithms at the same conditions. We used the DR-LVM algorithm instead of the M-LVM algorithm since they deliver the same result by almost all parameter combinations, furthermore, the DR-LVM and M-LVM algorithms will be compared in Section 7.5.5.

7.5.2. Basic Sampling Configuration

The basic sampling configuration of the tests is the cube (see Figure 7.5). The number of the points sampled inside the cube is 8000, whereby the distance between two neighbor

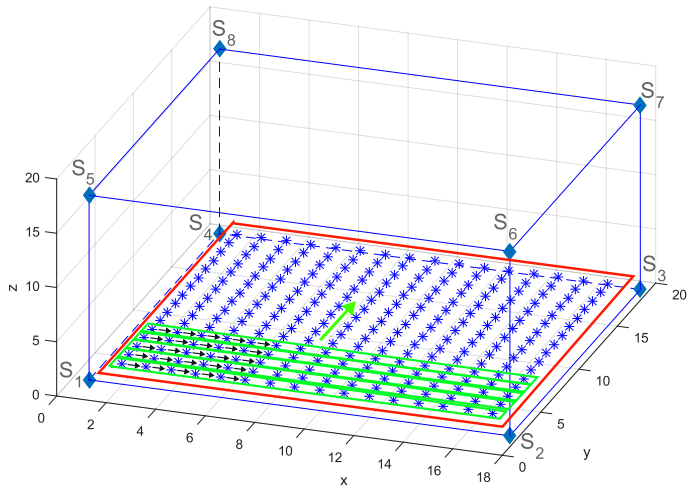


Figure 7.5.: *Layer view of sampled cube's space*

points (in the x- or y- direction) is one meter. The distance from the reference station to the nearby point is also one meter. The true positions are in fact unknown, but they are used in the test cases to evaluate the results. An approximate position is calculated for each true position by means of the multilateration algorithm as well as optimized by using the LVM algorithm. The multilateration algorithm uses the distances measured between the MS and the RSs. Furthermore, a PDOP value is calculated depending on the position of the references stations and the approximate position, which is a start point $\vec{x}^{(0)}$ for the LVM algorithm. The cube's space is sampled layer-by-layer from top to bottom, whereby each layer is sampled row-wise in the y-direction and each line is sampled from left to right (see Figure 7.5). The vertical green lines in Figure 7.5 illustrate the order in which the points are sampled at each layer marked by red lines.

7.5.3. Test Configuration with Eight Reference Stations

The reference stations $(S_1, S_2, \ldots, S_8)$ are placed at the corners of the cube to reach a good configuration (see Figure 7.6). The coordinates of the reference stations are defined as follows:

$$\begin{aligned} S_1(x_1, y_1, z_1) &= S_1(1, 1, 1) \\ S_2(x_2, y_2, z_2) &= S_2(19, 1, 1) \\ S_3(x_3, y_3, z_3) &= S_3(19, 19, 1) \\ S_4(x_4, y_4, z_4) &= S_4(1, 19, 1) \\ S_5(x_5, y_5, z_5) &= S_5(1, 1, 19) \\ S_6(x_6, y_6, z_6) &= S_6(19, 1, 19) \end{aligned} \tag{7.73}$$

$$S_7(x_7, y_7, z_7) = S_7(19, 19, 19)$$
$$S_8(x_8, y_8, z_8) = S_8(1, 19, 19)$$

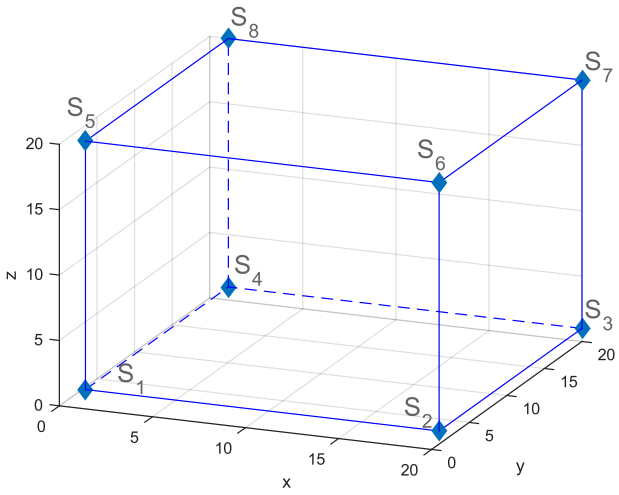


Figure 7.6.: *A 3-D configuration with 8 UWB reference stations*

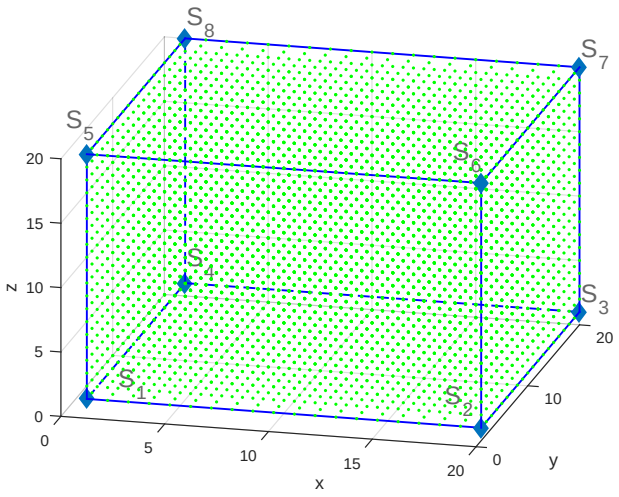


Figure 7.7.: *A 3-D configuration with 7992 positions searched*

The number of the points sampled inside the cube is 7992, since the reference stations are not incorporated (see Figure 7.7). Figure 7.8 depicts the PDOP values of the sampled points, whereas the vertical red lines represent the layers of the cube. Figure 7.9 is a section from Figure 7.8 and represents the PDOP values of the points inside a layer, which is illustrated in Figure 7.5.

The points between two vertical green lines in Figure 7.9 belong to a straight line, as illustrated by the horizontal green lines in Figure 7.5. The PDOP value of these points decreases the closer they are to the middle. In other words, the points inside the cube,

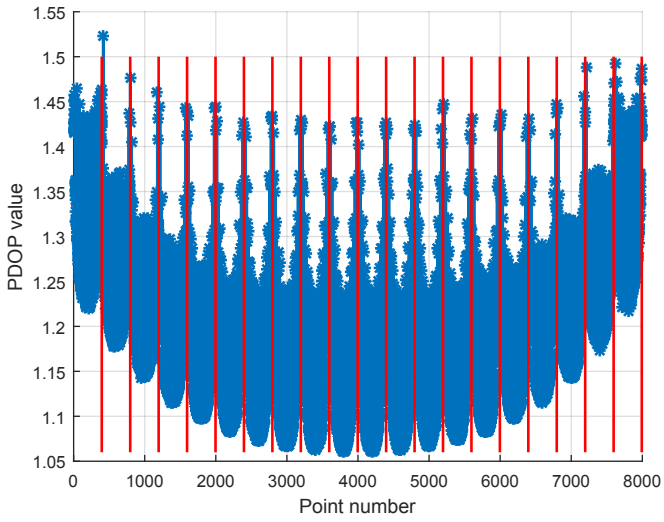


Figure 7.8.: *PDOP values as a function of the approximate (start) positions* $\vec{x}^{(0)}$

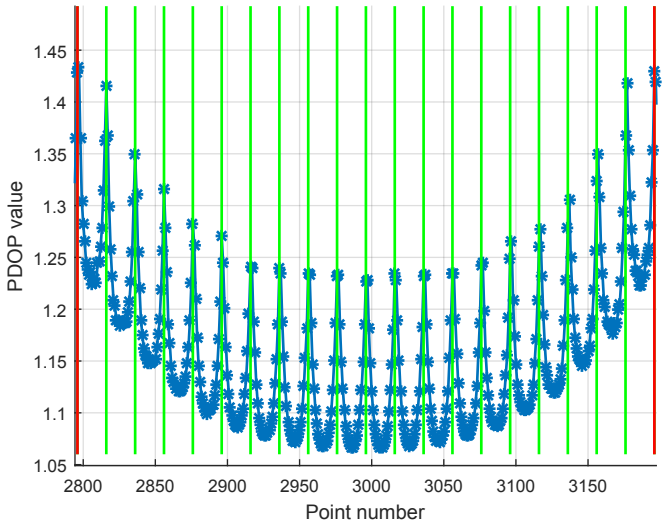
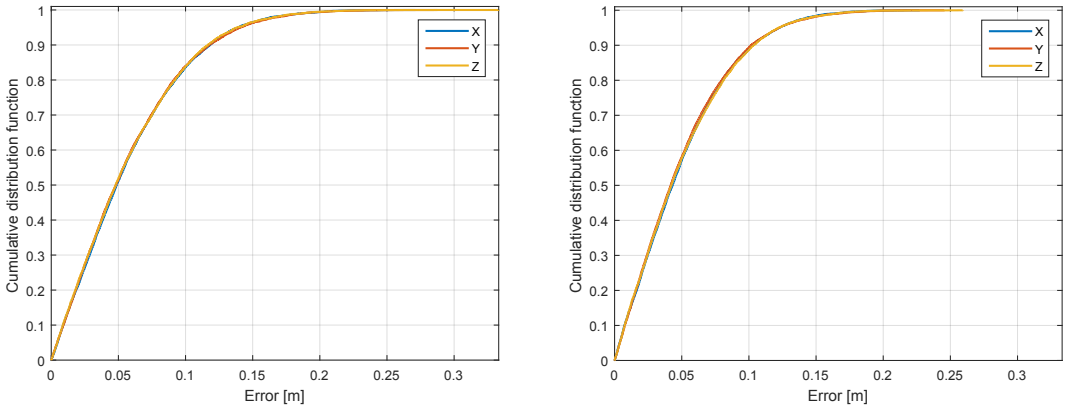


Figure 7.9.: *PDOP values of the points on a plan*

which are nearest to the reference stations, have a poorer PDOP value. In this case, the configuration of the reference stations with respect to the approximate position (start vector $\vec{x}^{(0)}$) is poor. The points in the middle of each row have the best PDOP values. The best PDOP value has the point located in the center of the cube. The PDOP values of the points inside a cube configuration with eight reference stations remain below 1.51.

Figure 7.10 shows a comparison between the multilateration and the DR-LVM algorithm, whereby the ECDF of the positions calculated with both algorithms are almost the same. The DR-LVM algorithm optimizes the position up to a maximum of 1.6 cm in all directions. In this case, the use of the LVM algorithm is not necessary.



(a) *ECDF of the points estimated by the ML algorithm.*

(b) *ECDF of the estimated positions after the use of the DR-LVM method.*

Figure 7.10.: *Comparison of the ML and the DR-LVM algorithms for a configuration with* 8 *anchors*

7.5.4. Test Configuration with Four Reference Stations

In this subsection, we investigate a practical case with four reference stations, which is the minimum number of reference stations in the 3-D case, otherwise the matrix $\boldsymbol{A}$ will be singular (*cf.* Equation (7.3)). We will treat a good as well as a poor configuration, which is formed by placing all the reference stations at almost the same height. The good and bad configurations are selected to demonstrate that the use of the LVM algorithm is not necessary by a good configuration. Furthermore, the bad configuration is formed to stress the algorithms and to demonstrate the necessity of the optimization algorithms. The optimization is achieved by the DR-LVM algorithm with $\tau = 10^{-6}$ and $\varepsilon = 10^{-2}$.

7.5.4.1. Four Reference Stations with Good Configuration

As shown in Figure 7.11, the reference stations are located pairwise at various heights. The points are sampled like the case for eight reference stations. The number of the sampled points is equal to 7996, since the RSs are excluded.

The coordinate of the reference stations $(S_1, \ldots, S_4)$ are defined as follows:

$$\begin{aligned} S_1(x_1, y_1, z_1) &= S_1(1, 1, 1) \\ S_2(x_2, y_2, z_2) &= S_2(20, 1, 20) \\ S_3(x_3, y_3, z_3) &= S_3(20, 20, 1) \\ S_4(x_4, y_4, z_4) &= S_4(1, 20, 20) \end{aligned} \tag{7.74}$$

The approximate positions and their PDOP values are calculated in the simulation. Figure 7.12 shows the PDOP values of approximate (not optimized) positions, which lie

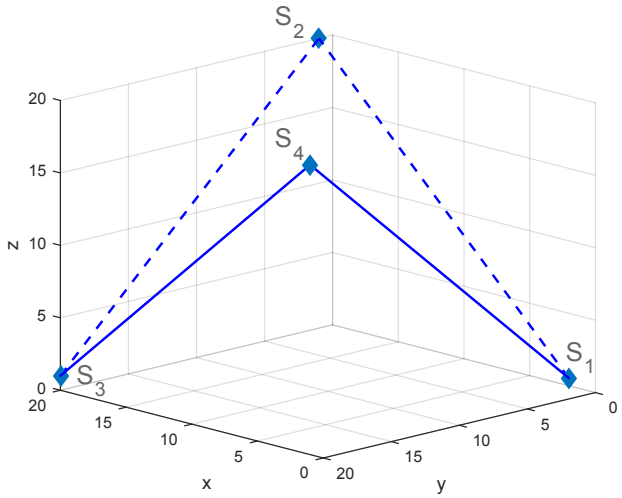


Figure 7.11.: *Good configuration with four reference stations*

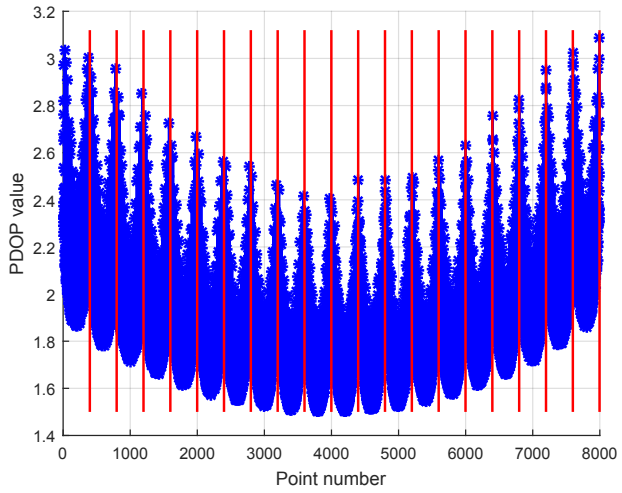


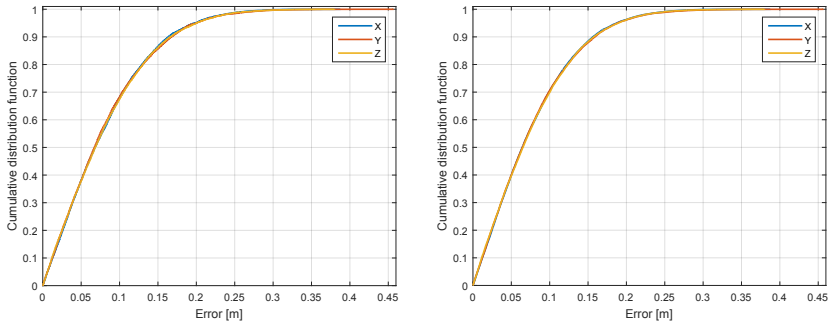
Figure 7.12.: *PDOP values of 4-RSs good configuration*

under 3.2.

The ECDF of the approximate positions before and after the use of the DR-LVM method is illustrated in Figure 7.13. The DR-LVM algorithm demonstrates only a minimal improvement of the position. Based on these and the previous results, the following rule can be extracted: The use of the LVM method is not necessary for PDOP values under 3.2.

7.5.4.2. Four Reference Stations with Bad Configuration

This configuration is composed of four reference stations, whereby the points are sampled in the same way as seen in the previous case, but the reference stations are approximately at the same height (see Figure 7.14). The z-coordinates of the RS_1 up to RS_4 are equal to 1, 1.3, 1.5 and 1.9, respectively. The x- and y-coordinates of the RSs are the same as defined



(a) *ECDF of the points estimated by the ML algorithm.*

(b) *ECDF of the points estimated by the DR-LVM algorithm.*

Figure 7.13.: *Comparison of the ML and the DR-LVM algorithms for a good configuration with 4 anchors*

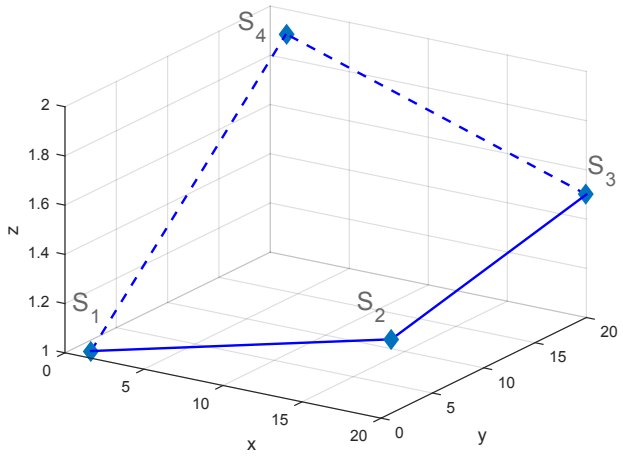


Figure 7.14.: *Bad configuration with four reference stations*

in (7.74).

The approximate positions and their PDOP values are calculated for each point. The approximate position is calculated by using the multilateration method as well as noisy distances. The PDOP values illustrated in Figure 7.15 are a function of the calculated approximate positions. These positions are the start vectors for the LVM algorithm. In Figure 7.15, the PDOP values are significantly larger compared to the previous test scenario with good configuration and four anchors. The PDOP values are greater than 5.

The ECDFs of the points calculated by the multilateration method and optimized by the DR-LVM algorithm are illustrated in Figures 7.16 and 7.17.

The absolute errors for 90 % of the simulated points before running the DR-LVM algorithm in the x- and y-directions are 0.17 m and 0.30 m (see Figure 7.16(a)). The absolute errors in the x- and y-directions are limited to 0.16 m and 0.18 m after using the DR-LVM method, as depicted in Figure 7.16(b).

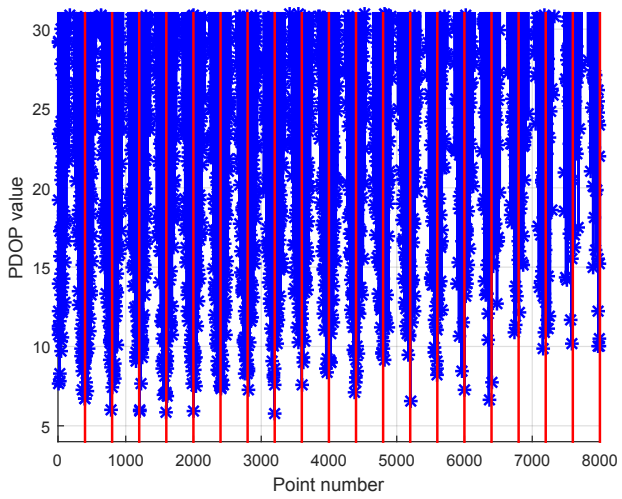
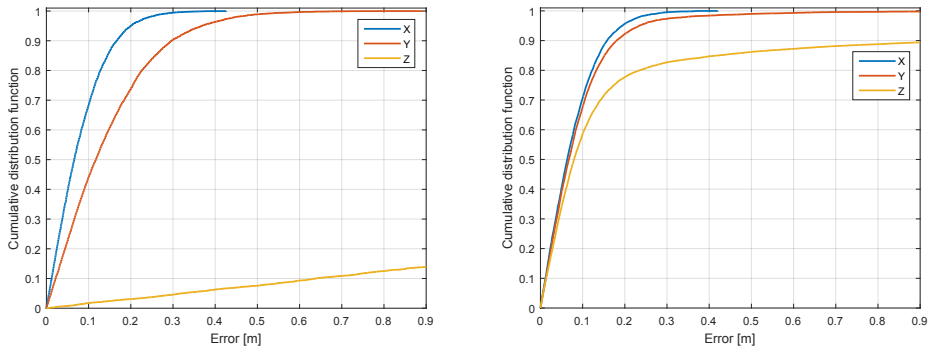


Figure 7.15.: *PODP values of a 4-RSs bad configuration*



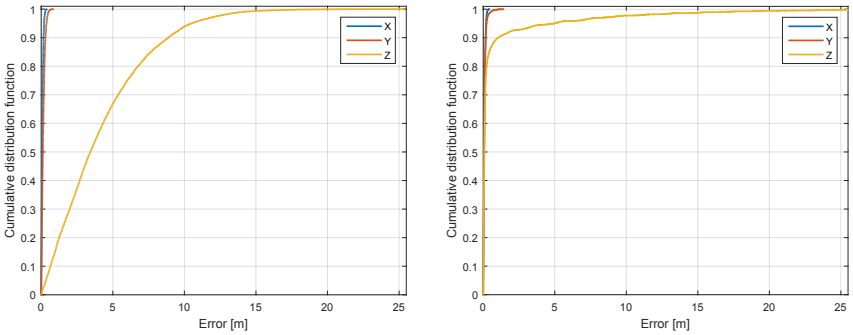
(a) *ECDF of the ML method.* **(b)** *ECDF of the LVM method.*

Figure 7.16.: *Bad configuration with 4 anchors, focus on the x and y errors*

The absolute errors of the optimized positions are significantly reduced using the DR-LVM algorithm, especially in the z-direction. The errors in the z coordinates are limited to 0.2 m for 80 % of the points, while they amount to about 7 m before the running of the DR-LVM algorithm (*cf.* Figures 7.17(a) and 7.17(b)). The second rule, which can be extracted is as follows: The invocation of the DR-LVM algorithm is indispensable for PDOP values above 8. The PDOP value is set to 8 as a limit, since the PDOP value of the 99.63% of the points is about 8.

7.5.4.3. Parameter Analysis

The choice of the τ and the ε parameters is very important for the efficiency and the success rate of the LVM algorithm. Therefore, various test cases are performed to determine the appropriate parameter combination. Figure 7.18 shows the results of the test cases of the DR-LVM algorithm implemented with different precision ε and τ values.



(a) *ECDF of the ML method.* **(b)** *ECDF of the LVM method.*

Figure 7.17.: *Bad configuration with 4 anchors, focus on the z errors*

As can be seen from Figure 7.18, the absolute error in the x-, y- and especially in the z-direction will be smaller by the same τ value, the higher is the claimed accuracy (ε). That means, small absolute errors require small ε values. By the same accuracy value (ε), τ with the value of 10^{-6} delivers the best result (see Figures 7.18). The absolute position error remains unchanged with precision values less than or equal to 10^{-3} ($\varepsilon \leq 10^{-3}$) for all τ values. For the precision $\varepsilon = 10^{-2}$, the LVM approach has the same absolute position error except for τ equal to 1. For the precision $\varepsilon = 10^{-1}$, the LVM algorithm shows poor results compared with ε equal to 10^{-6} up to 10^{-3}.

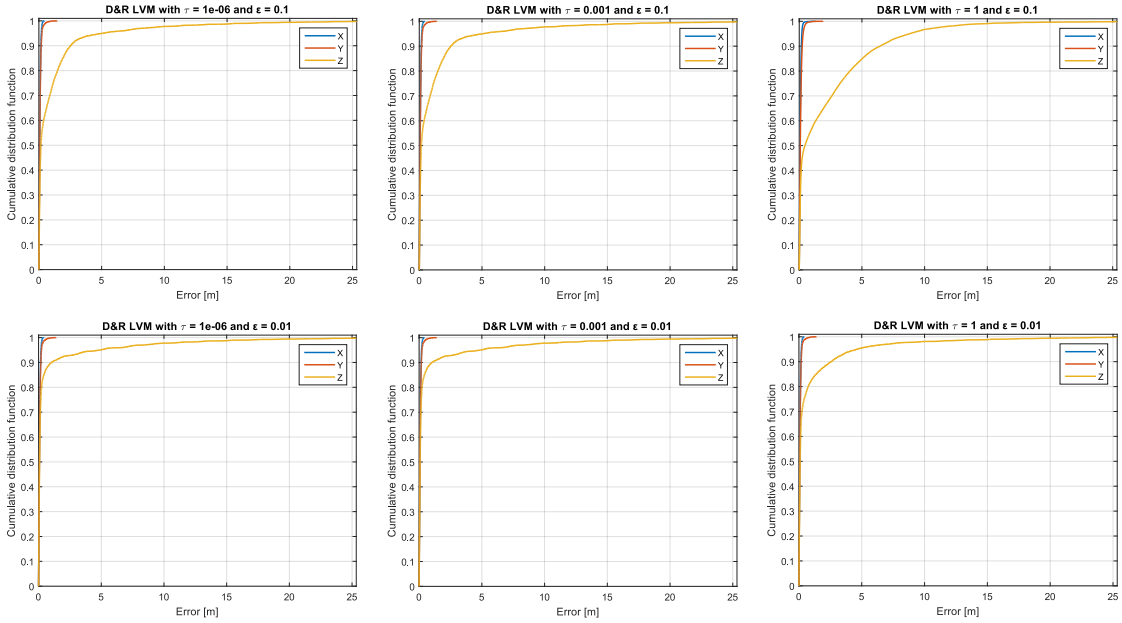


Figure 7.18.: *Cont.*

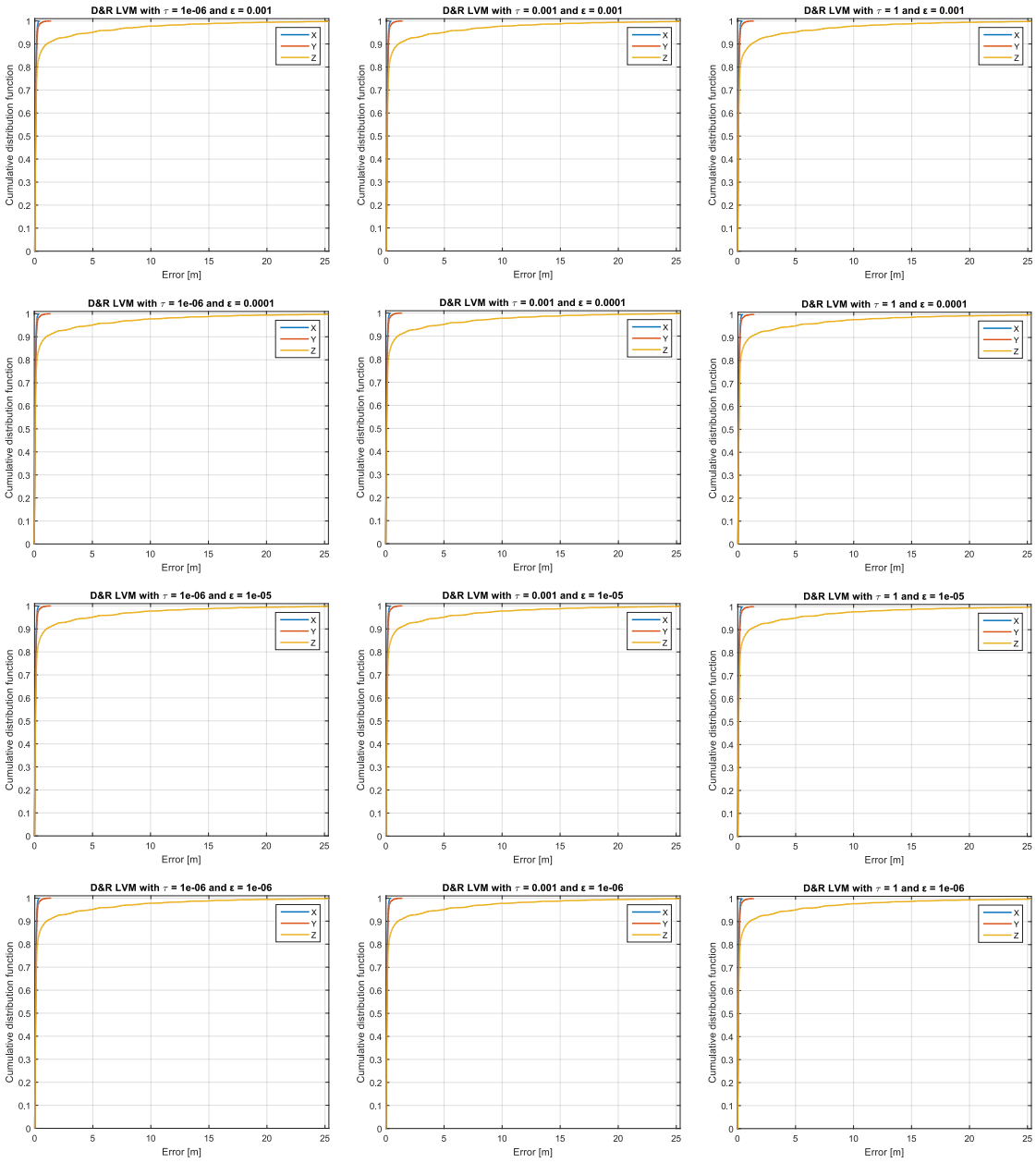
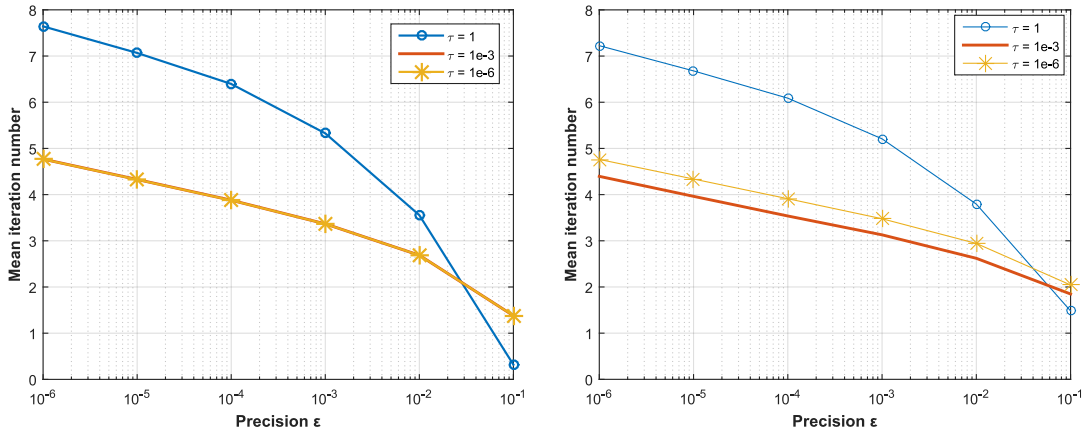


Figure 7.18.: *Comparison of the DR-LVM algorithm with six accuracy bounds (ε)*

7.5.4.4. Iteration Number Analysis

The iteration number of the LVM algorithm will be analyzed, in order to define the best (ε, τ) combination based on the previous observations. Figure 7.19(a) shows the mean iteration number of the DR-LVM algorithm as a function of the precision ε for τ equal to 1, 10^{-3}, and 10^{-6}. In this case, the DR-LVM algorithm with τ equal to 1 shows the largest iteration number. In contrast, the DR-LVM algorithm needs the smallest as well as approximately



(a) *DR-LVM algorithm. The curves of* τ *equal* $1e^{-3}$ *and* $1e^{-6}$ *overlap.*

(b) *M-LVM algorithm.*

Figure 7.19.: *Mean iteration numbers of the DR- and M-LVM algorithms for* $\tau = 1, 10^{-3}, 10^{-6}$

the same mean number of iterations for $\tau = 10^{-3}$ and $\tau = 10^{-6}$.

The DR-LVM algorithm, in general, requires for $\tau = 10^{-3}$ more iterations as for $\tau = 10^{-6}$. The mean iteration number decreases with increasing ε values; therefore, the ε-parameter is selected for the largest value at which the absolute position error remains unchanged. As previously demonstrated, this value is equal to $\varepsilon = 10^{-2}$.

The third rule that can be extracted from the parameter analysis is: By the use of the DR-LVM algorithm, $\varepsilon = 10^{-2}$ and $\tau = 10^{-6}$ are to choose, since they deliver the best accuracy and smallest iteration number.

7.5.5. Comparison of Dahmen-Reusken and Madsen LVM–Algorithms

Both variants of the LVM algorithms are compared with different precision ε- and τ- values. Therefore, the DR-LVM and M-LVM methods are performed with the following parameter values: ε from 10^{-1} to 10^{-6} and τ equal to 1, 10^{-3}, or 10^{-6}.

The DR- and M-LVM algorithms have almost the same iteration numbers. The DR-LVM algorithm requires a little fewer iteration numbers as the M-LVM algorithm for τ equal to 10^{-6}, but no general statement can be made about other τ-values (cf. Figure 7.19).

The DR-LVM and M-LVM methods show different results by $\varepsilon = 10^{-1}$ (see Figure 7.20). They reveal almost the same result by $\varepsilon = 10^{-2}$ except for τ equal to 1. Both algorithms deliver the same result independently from τ values for ε equal to 10^{-6} up to 10^{-3}.

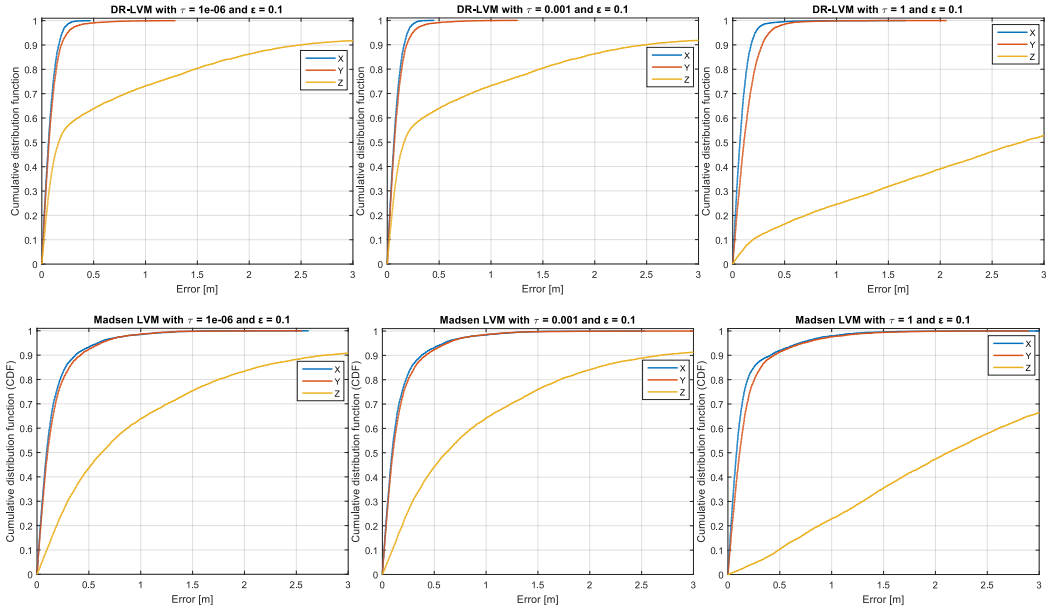


Figure 7.20.: *ECDF of the Dahmen-Reusken and Madsen algorithms with* $\varepsilon = 10^{-1}$

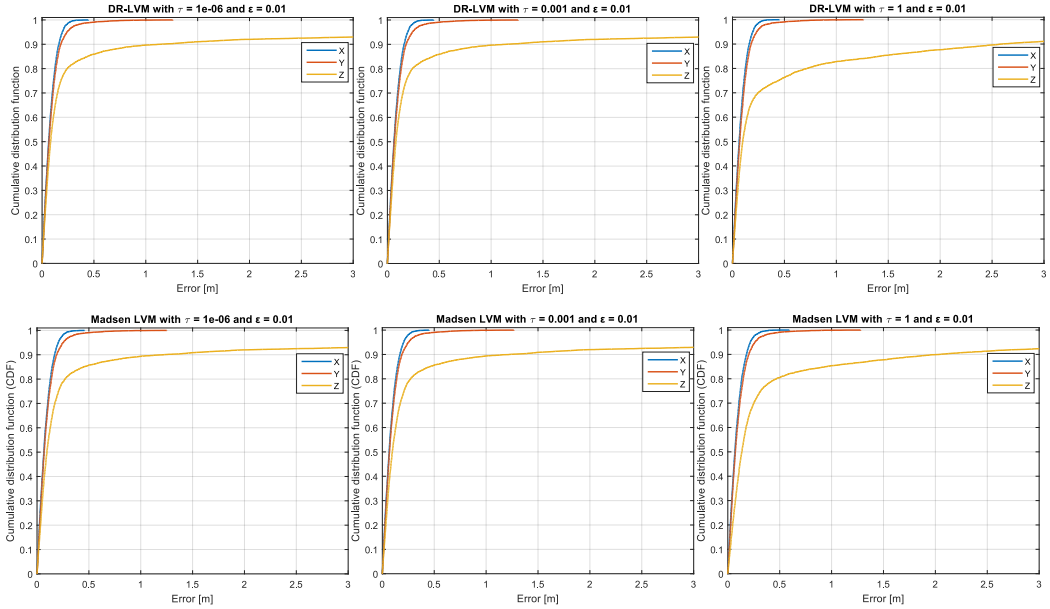


Figure 7.21.: *ECDF of the Dahmen-Reusken and Madsen algorithms with* $\varepsilon = 10^{-2}$

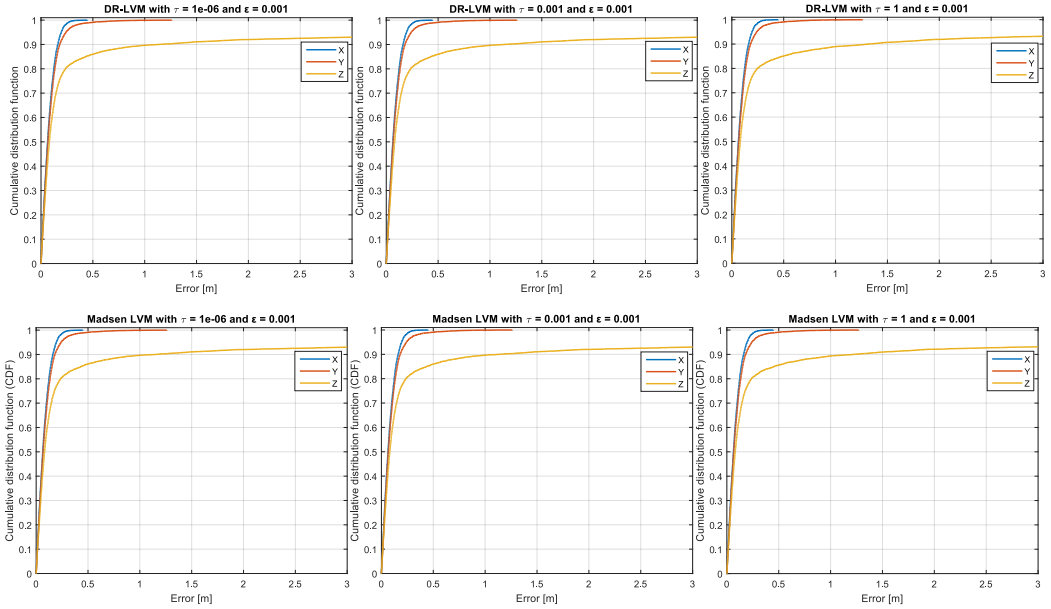


Figure 7.22.: *ECDF of the Dahmen-Reusken and Madsen algorithms with* $\varepsilon = 10^{-3}$

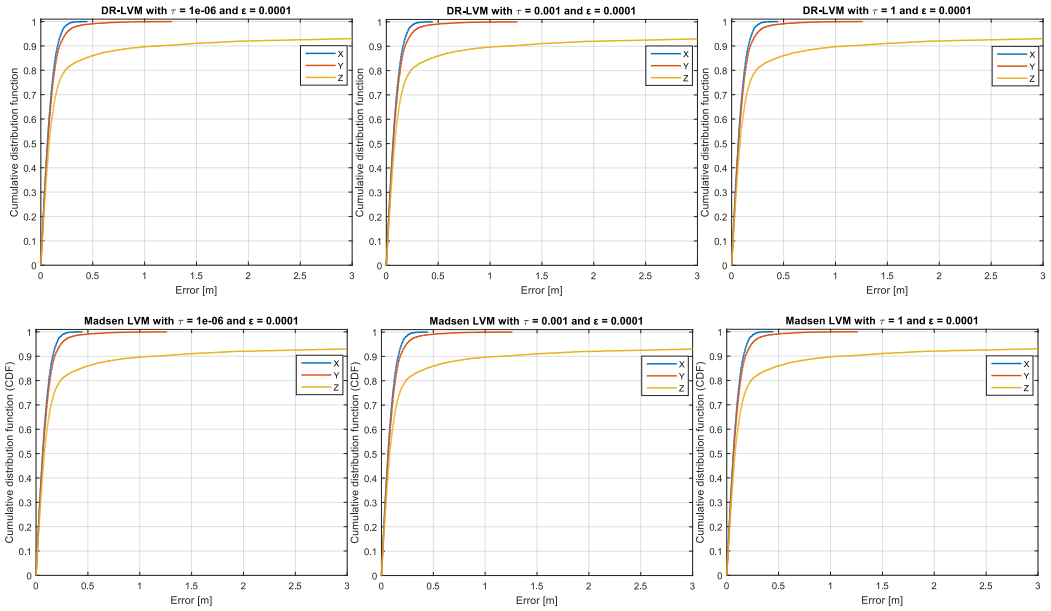


Figure 7.23.: *ECDF of the Dahmen-Reusken and Madsen algorithms with* $\varepsilon = 10^{-4}$

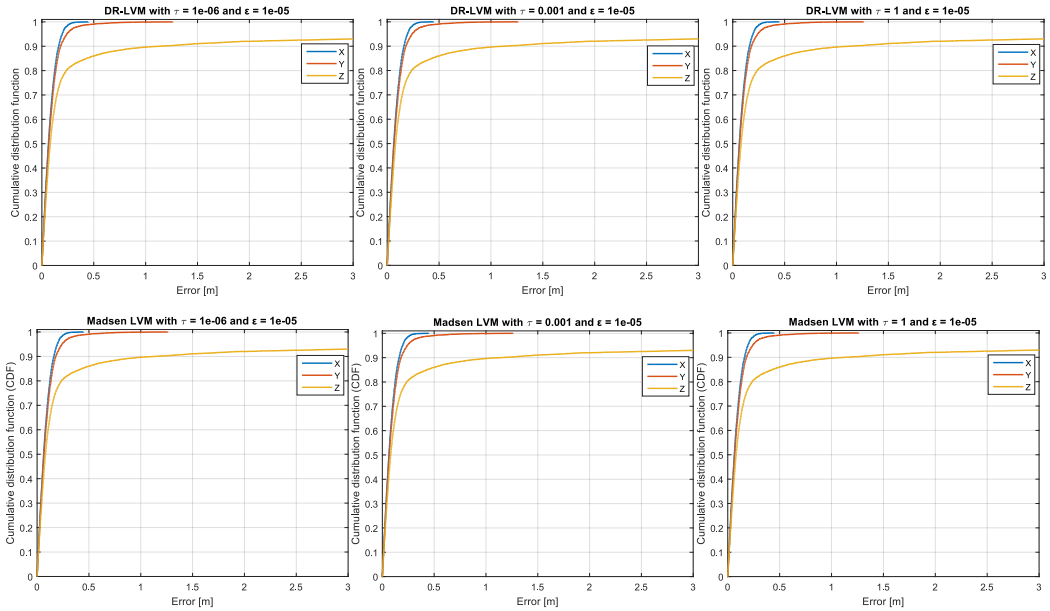


Figure 7.24.: *ECDF of the Dahmen-Reusken and Madsen algorithms with* $\varepsilon = 10^{-5}$

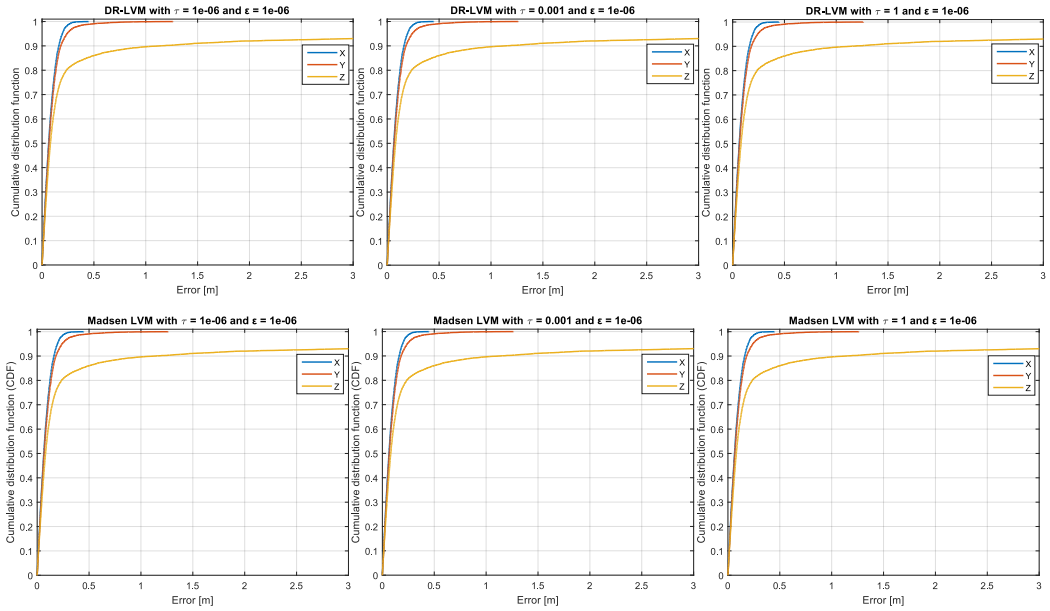
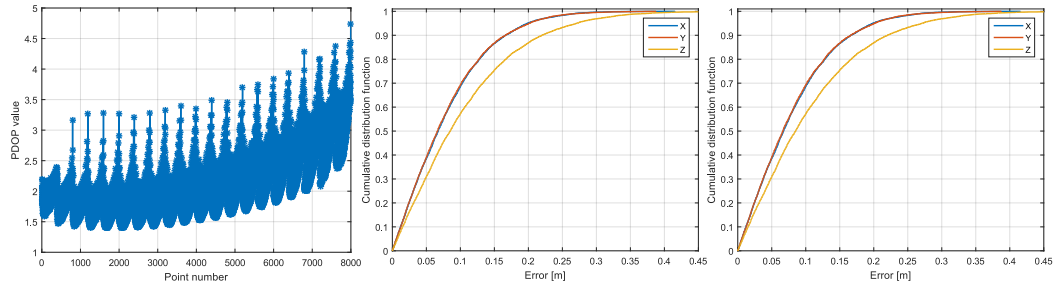


Figure 7.25.: *ECDF of the Dahmen-Reusken and Madsen algorithms with* $\varepsilon = 10^{-6}$

7.5.6. Test Scenario with Increasing Number of Reference Stations

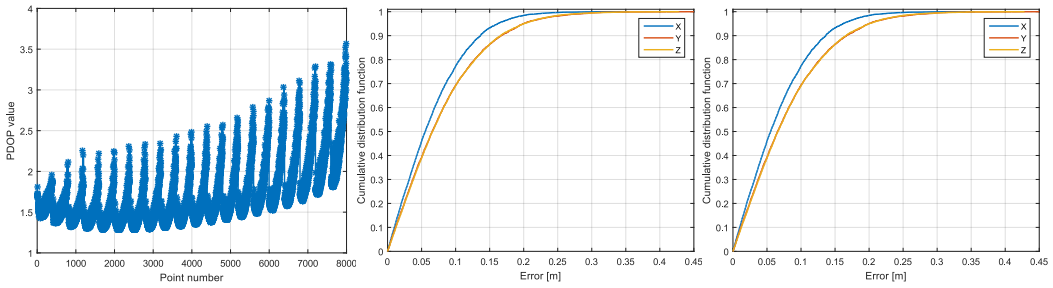
We perform various tests to demonstrate that the correlation between PDOP and the position errors stays by RS-configurations different than the previous eight- and four-RS configuration. These tests also revealed that the position accuracy depends not only on the location of the

reference stations to the approximate positions, but also on the number of the deployed reference stations. Therefore, a test scenario is created to demonstrate the impact of the number of the RSs on the position accuracy. The number of the RSs varies from five up to eight. The coordinates of the RSs (S_1 up to S_8) used are defined in (7.73). The PDOP values become smaller with increasing number of the RSs, as shown in Figures 7.26(a), 7.27(a), 7.28(a), and 7.29(a). The maximal PDOP value is below 4.6 for all anchor configurations, which refers to a good room configuration. The Figures 7.26(b), 7.27(b), 7.28(b), and 7.29(b) show that the absolute error of the approximate positions decreases with increasing number of the RSs. The maximum improvement of the approximate positions varies from 2 cm up to 4 cm, as is shown in Figures 7.26(c), 7.27(c), 7.28(c), and 7.29(c).



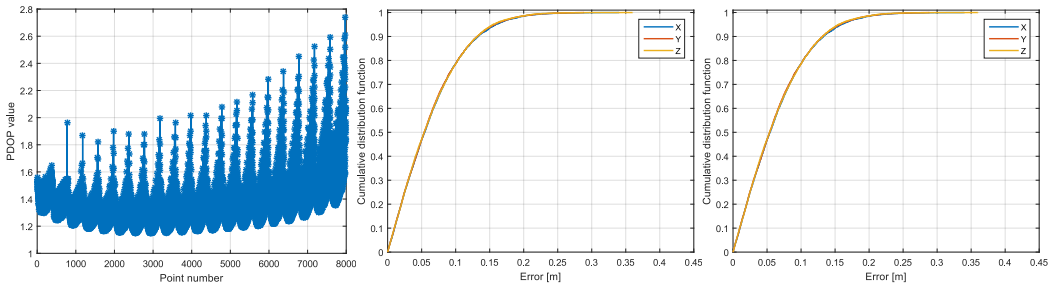
(a) **(b)** **(c)**

Figure 7.26.: *5-anchor configuration*



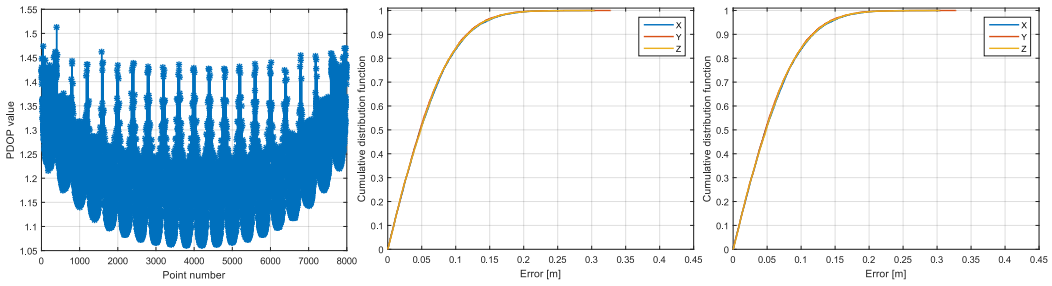
(a) **(b)** **(c)**

Figure 7.27.: *6-anchor configuration*



(a) **(b)** **(c)**

Figure 7.28.: *7-anchor configuration*



(a) **(b)** **(c)**

Figure 7.29.: *8-anchor configuration. (**a**) PDOP value, (**b**) ECDF before the use of the LVM algorithm, (**c**) ECDF after the use of the LVM algorithm*

7.5.7. Evaluation of the Multipath Distance Detection and Mitigation and Position Optimization Algorithm

We perform a simulation experiment by extending the simulation in Section 7.5.3 with the multipath effect to validate the performance of the MDDM algorithm. The multipath effect is modeled as follows [219]:

$$\epsilon_{M,W}(d) = \gamma_W \log(1+d) \tag{7.75}$$

$$= G(m_{M,W}, \sigma_{M,W})\, log(1+d) \tag{7.76}$$

$$f_{\gamma \mathrm{w}}(x) = \frac{1}{\sigma_{M,W}\sqrt{2}} \mathrm{e}^{-\frac{\left(x-m_{M,W}\right)^2}{2\sigma^2_{M,W}}}, \tag{7.77}$$

where, $G(m_{M,W}, \sigma_{M,W})$ is a Gaussian random variable with mean $m_{M,W}$ and variance $\sigma^2_{M,W}$. The parameters used for the simulation are summarized in Table 7.2.

Table 7.2.: *Parameter used for the evaluation of the multipath distance detection and mitigation algorithm*

Parameter	Value
β_0	0.2
β_1	0.8
Maximal iteration number	100
ε	$1e^{-5}$
τ	$1e^{-6}$
PDOP Threshold	8
Number of multipath affected distances	2
$m_{M,W}$	0.88
$\sigma_{M,W}$	152.2

Figure 7.30 shows the ECDF of the position errors without and with the use of the MDDM algorithm. The position error is less than 3.2 m for 90 % of the points before the use of

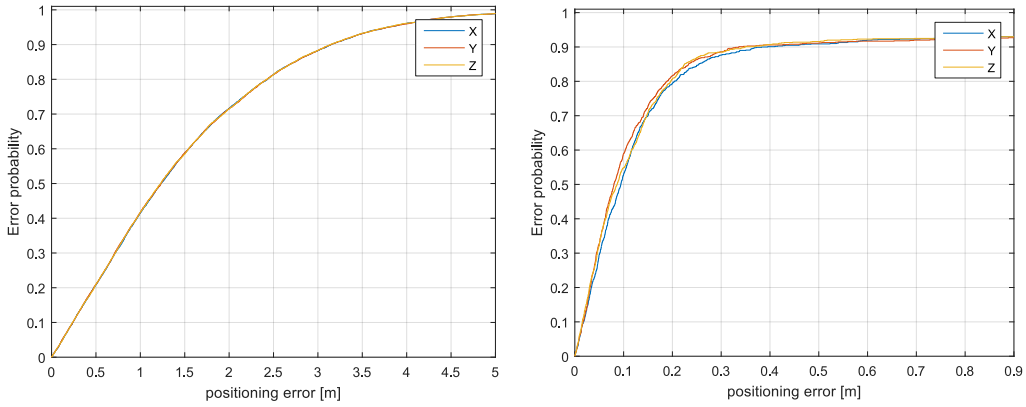


Figure 7.30.: *Empirical CDFs. (**a**) Without the use of the MDDM algorithm, (**b**) with the use of the MDDM algorithm. MDDM, Multipath Distance Detection and Mitigation*

MDDM (*cf.* Figure 7.30(a)). In contrast, the position error is reduced to 40 cm for 90 % of the points after applying the MDDM method (*cf.* Figure 7.30(b)).

7.6. Evaluation of UWB-ILS in a Real Environment

In this section, the DR-LVM algorithm is deployed in a real-world scenario, whereby four RSs are used. The bad configuration is formed to stress the algorithms. The DR-LVM algorithm is evaluated in respect to accuracy and iteration number to validate the rules extracted from the previous Section 7.5. Furthermore, the computing time of the MDDM algorithm is measured on resource-limited devices. Finally, the DR-LVM method is compared with the GN algorithm.

Four reference stations are used, as shown in Figure 7.31, whereby their coordinates are defined as follows:

$$\begin{aligned} S_1(x_1, y_1, z_1) &= S_1(0, 0, 1.67) \\ S_2(x_2, y_2, z_2) &= S_2(4.5, 0, 0.75) \\ S_3(x_3, y_3, z_3) &= S_3(4.5, 4.5, 0.75) \\ S_4(x_4, y_4, z_4) &= S_4(0, 4.92, 0.86) \end{aligned} \tag{7.78}$$

Four UWB reference transceivers were in a 6 m × 7 m room and the UWB mobile stations are placed in several points and at three different heights in the room. The location of the MS is measured at 27 different locations, whereby the measurement is repeated fifty-three times at each location (see Figure 7.32).

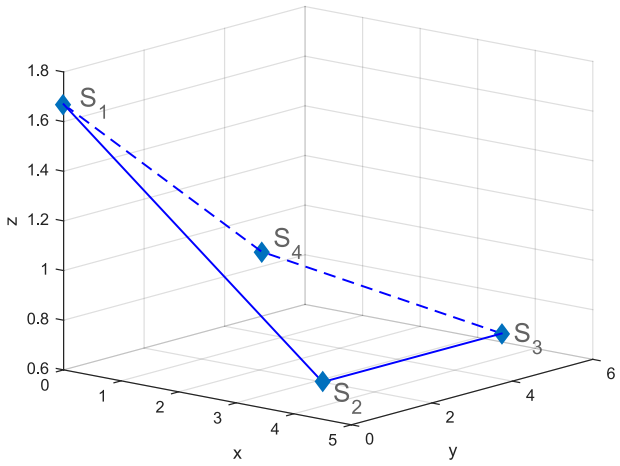


Figure 7.31.: *A 4-anchor configuration for a real scenario analysis*

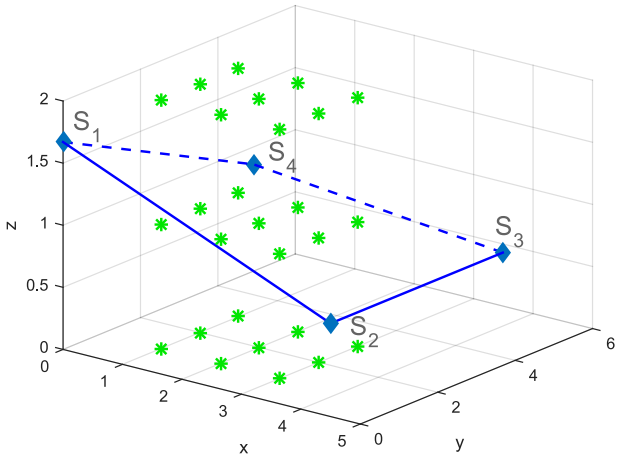


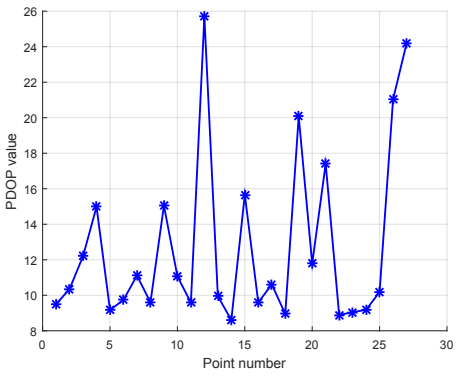
Figure 7.32.: *A 4-anchor configuration with 27 points*

7.6.1. Accuracy Measurement

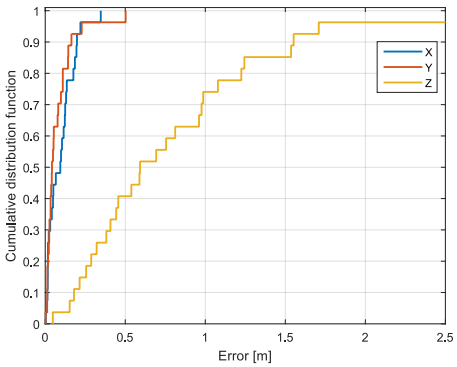
The PDOP values, which are over 8.6, are calculated depending on the location of the approximate positions as well as of the RSs, as shown in Figure 7.33(a). The DR-LVM algorithm is performed with $\tau = 10^{-6}$ and $\varepsilon = 10^{-2}$ according to the third rule derived in Section 7.5.4. The Figures 7.33(b) and 7.33(c) show that the DR-LVM algorithm improves the approximate position in all directions, particularly in the z-direction. The true positions are surveyed by using a laser-based system with a precision of less than 5 mm.

7.6.2. Iteration Number

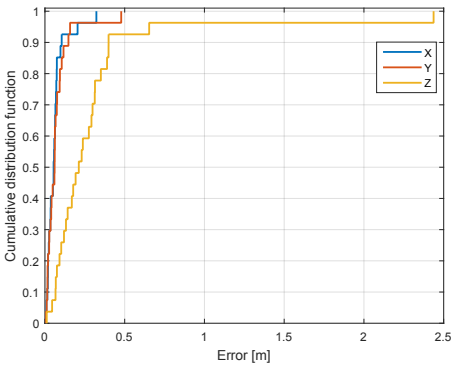
In this experiment, we chose (ε, τ) equal to $(10^{-2}, 10^{-6})$ to reach the smallest iteration number with a minimum of position errors, as demonstrated in Section 7.5.4.4. Figure 7.34 shows that (ε, τ) equal to $(10^{-2}, 10^{-6})$ delivers the smallest iteration number by the minimal



(a) *PDOP values*



(b) *ECDF before the use of the LVM algorithm*



(c) *ECDF after the use of the LVM algorithm*

Figure 7.33.: *PDOP values and the comparison of the ML method with the LVM algorithm. ML, Multilateration, LVM, Levenberg–Marquardt. ECDF, Empirical Cumulative Distribution Function*

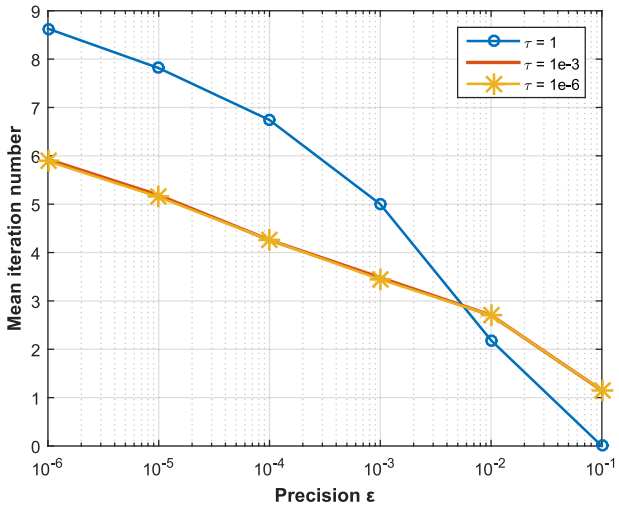
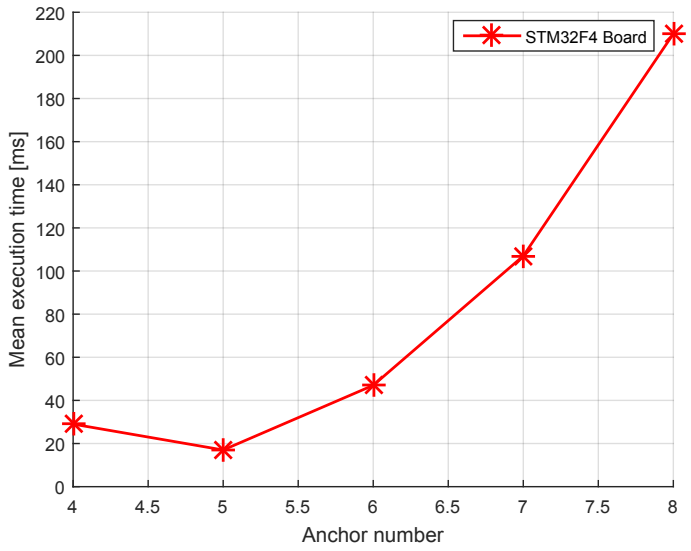


Figure 7.34.: *Iteration number of the DR-LVM algorithm in real scenario. The curves of τ equal $1e^{-3}$ and $1e^{-6}$ overlap*

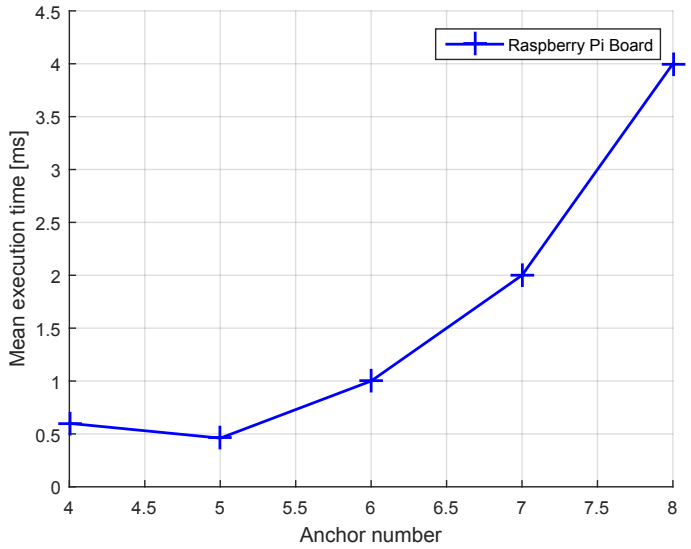
errors for the measured points, which confirms the results found in Section 7.5.4.3 and 7.5.4.4.

7.6.3. Computing Time Measurement

The computing time of the MDDM algorithm is evaluated on STM32F4-MCU as well as on Raspberry Pi 3 [220]. We used Raspberry Pi 3, which has more capacity (Quad Core 1.2 GHz and 1 GB RAM) as the STM32F4-MCU to demonstrate the scalability of the MDDM algorithm. By a configuration of 8-RSs, the MDDM algorithm requires approximately 210 ms and 4 ms on STM32F4-MCU and Raspberry Pi 3. Figure 7.35 shows the mean execution time of the MDDM approach as a function of the anchor numbers.



(a) *Time Measurement on STM32F4 MCU*

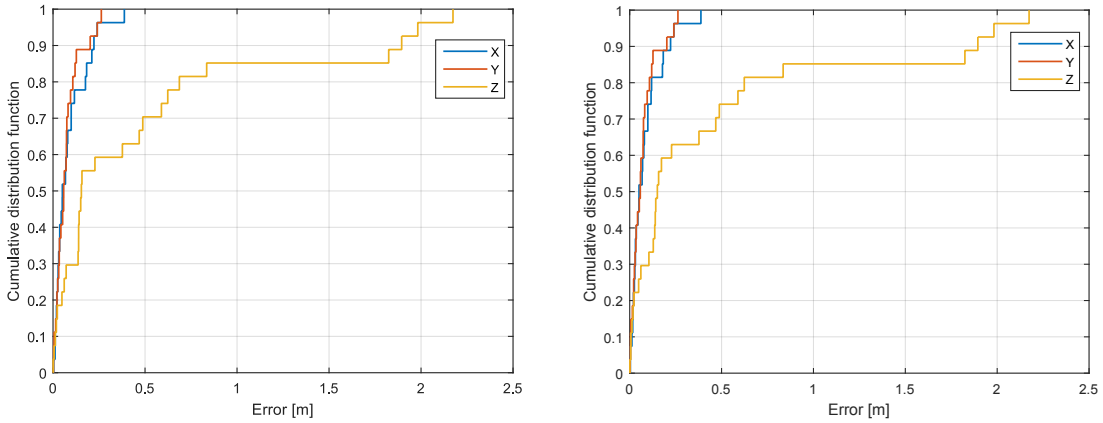


(b) *Time Measurement on Raspberry Pi 3*

Figure 7.35.: *Computing time evaluation of the MDDM algorithm*

7.6.4. Comparison of the Levenberg–Marquardt and Gauss–Newton Method

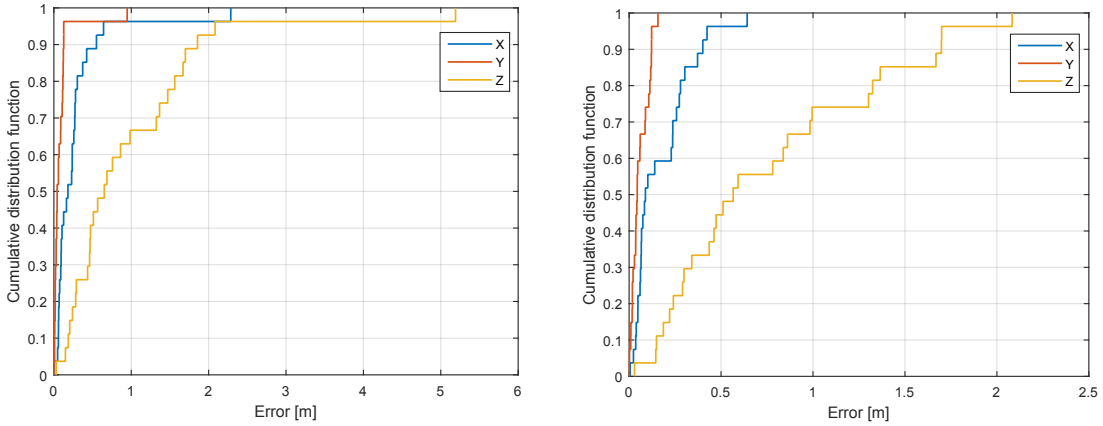
The DR-LVM algorithm will be compared with the Gauss–Newton method by using ε equal to 10^{-5}. Both algorithms are compared based on real measurements within the same room as in Figure 7.31. The LVM and GN methods show almost the same performance if the quadratic matrix $\boldsymbol{J}_f^T \boldsymbol{J}_f$ is not singular, as shown in Figure 7.36.



(a) *ECDF of the GNM.* **(b)** *ECDF of the LVM algorithm.*

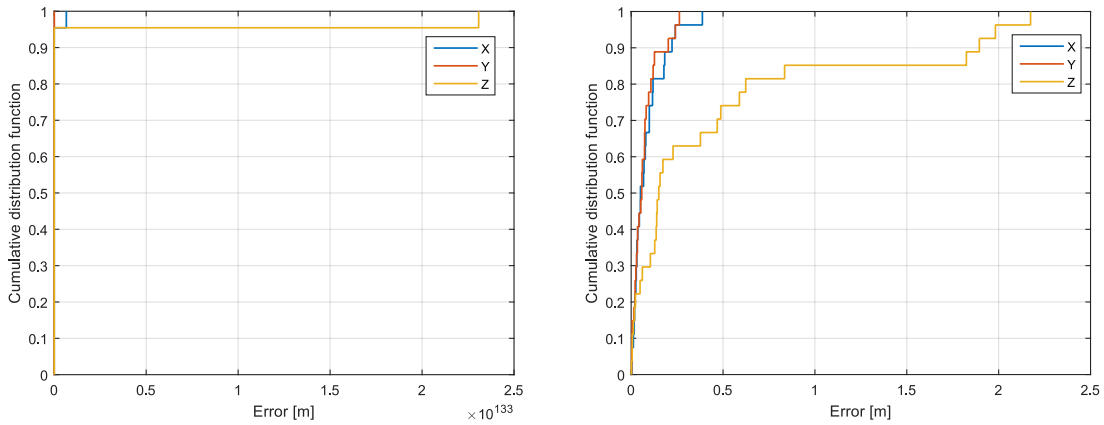
Figure 7.36.: *Comparison of the GNM and LVM algorithm if the matrix $\boldsymbol{J}_f^T \boldsymbol{J}_f$ is not singular*

The GNM shows a poor performance as well as diverges, when the matrix $\boldsymbol{J}_f^T \boldsymbol{J}_f$ is near singular. In contrast, the LVM algorithm remains stable (see Figure 7.37).



(a) *ECDF of the GNM.* **(b)** *ECDF of the LVM algorithm.*

Figure 7.37.: *Comparison of the GNM and LVM method if the matrix $\boldsymbol{J}_f^T \boldsymbol{J}_f$ is near-singular*



(a) *ECDF of the GNM.* **(b)** *ECDF of the LVM algorithm.*

Figure 7.38.: *Comparison of the GNM and LVM algorithm if the matrix $\boldsymbol{J}_{\boldsymbol{f}}^{T}\boldsymbol{J}_{\boldsymbol{f}}$ is singular*

The GN algorithm fails in case the matrix $\boldsymbol{J}_{\boldsymbol{f}}^{T}\boldsymbol{J}_{\boldsymbol{f}}$ is singular, while the LVM algorithm remains stable, as illustrated in Figure 7.38. The mean iteration numbers of the GN and the LVM methods are equal to 5 and 3, respectively, if the matrix $\boldsymbol{J}_{\boldsymbol{f}}^{T}\boldsymbol{J}_{\boldsymbol{f}}$ is not singular. The mean iteration of the GN method increases to the maximal iteration number (100), if the matrix $\boldsymbol{J}_{\boldsymbol{f}}^{T}\boldsymbol{J}_{\boldsymbol{f}}$ is near-singular or singular; while the mean iteration number of the LVM method stays stable and rises to approximately 11 iterations. The GN and LVM algorithms require an average time of 1561 μs and 1548 μs per iteration, respectively, by using an MCU running at 168 MHz.

7.7. Evaluation of MILPS in a Simulated Environment

The LVM algorithm is analyzed for the MILPS by using the PDOP values and the ε and τ parameters, as seen in the previous Section 7.5. The tests are performed in the same way as in the previous Sections 7.5.3 and 7.5.4, whereas the DR-LVM is used. The approximate position is calculated by using noisy field strengths. The MILPS has a reach of about 9 m, whereby a configuration with eight coils as reference stations is illustrated in Figure 7.39 [221].

7.7.1. PDOP Analysis

There is no correlation between the absolute errors and the PDOP values by MILPS. The LVM method is needed, although the eight-reference configuration in a cube form enables good coverage. The PDOP values vary from 1.623 to 5 for the most approximate (start) positions. The use of the LVM approach is indispensable since the absolute errors in all direction are approximately 1.5 m for 90 % of the points. The use of the LVM approach limits the errors to about 40 cm for 90 % of the points, as shown in Figure 7.40 [221].

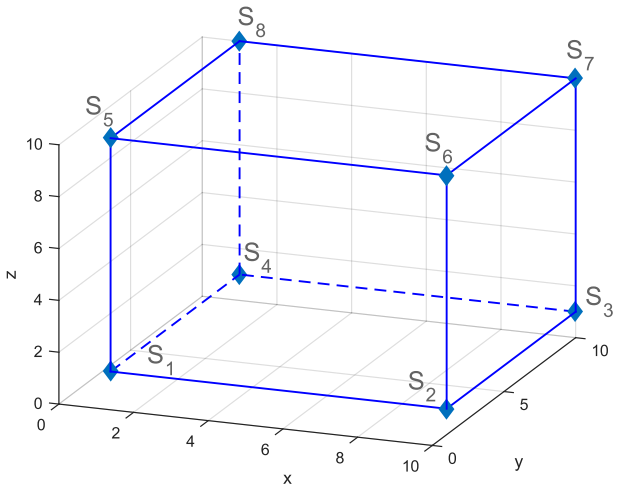
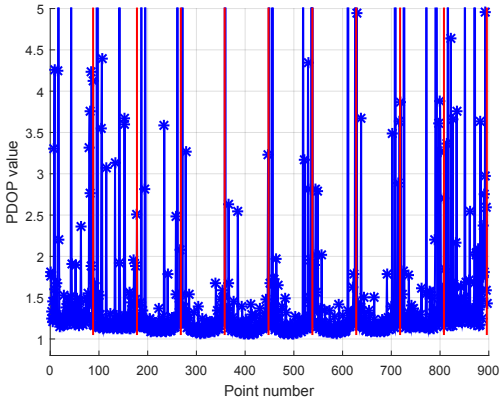
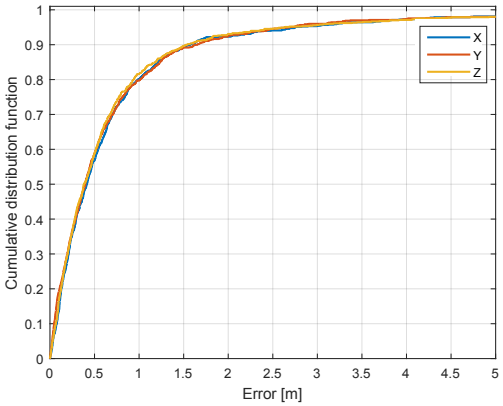


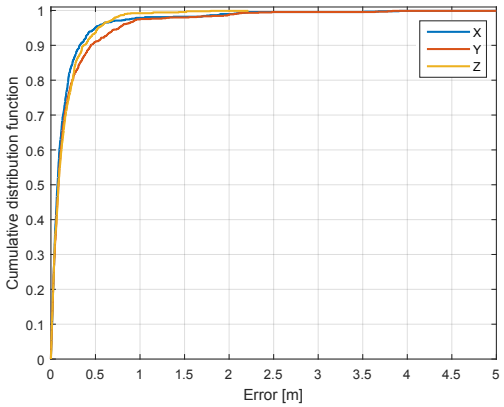
Figure 7.39.: *MILPS: a 3-D configuration with 8 coil configuration*



(a) *PDOP values*



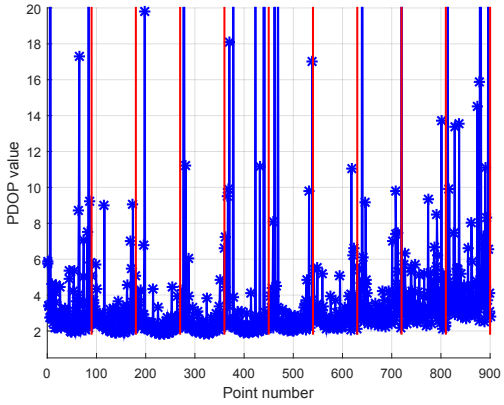
(b) *ECDF of the points estimated by the ML algorithm.*



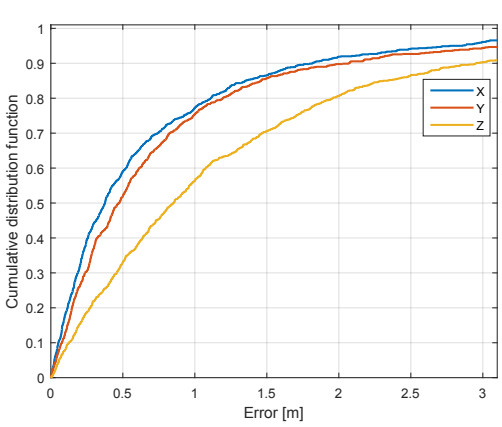
(c) *ECDF of the estimated positions after the use of the LVM algorithm.*

Figure 7.40.: *PDOP analysis of a configuration with 8 reference stations*

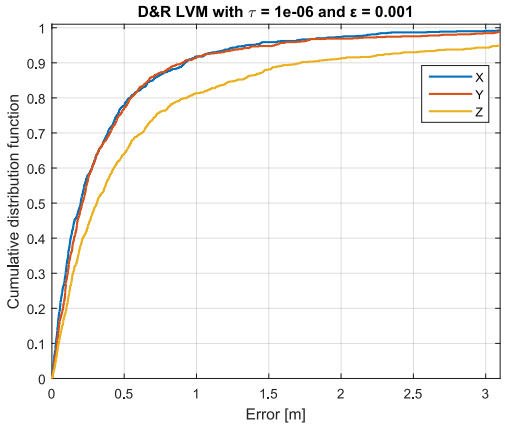
Furthermore, there are no clear boundaries for the PDOP values as in the previous case of the UWB-ILS (*cf.* Section 7.5). The values of the PDOP of the configuration with 8 references stations overlap with the values of the configuration with 4 references stations, as shown in Figures 7.40(a) and 7.41(a). The estimated positions in the configuration with 4 RSs shows more errors as in the configuration with 8 RSs, therefore the LVM method is also needed (see Figure 7.41).



(a) *PDOP values*



(b) *ECDF of the points estimated by the ML algorithm.*



(c) *ECDF of the estimated positions after the use of LVM algorithm.*

Figure 7.41.: *PDOP analysis of a configuration with 4 reference stations*

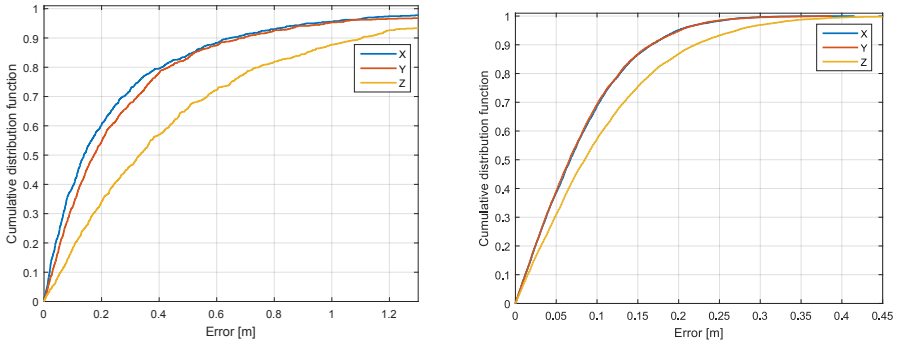
The LVM algorithm is necessary to improve the approximate positions by MILPS due to the nonlinear measurement model. In addition, the errors by MILPS are more complex than by the UWB-based ILS, since the measurements are influenced by ambient noise.

7.7.2. Noise Analysis

The ambient noise, as well as the measurement inaccuracies of the sensors, also affect the position accuracy of the approximate positions calculated by the multilateration method.

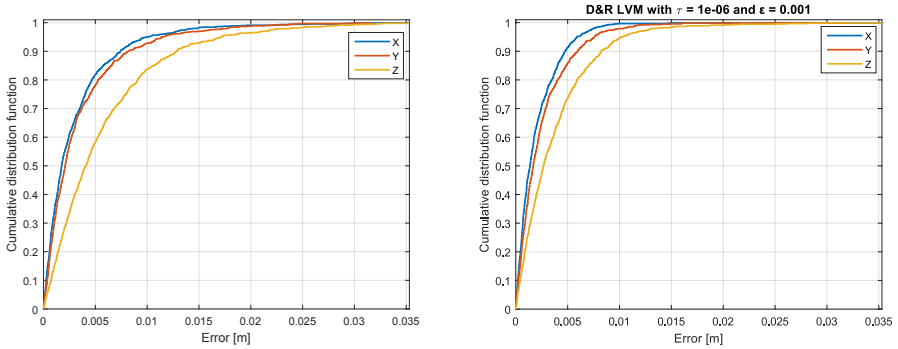
The LVM algorithm reaches an improvement of the position error in the meter as well as

in the centimeter range for noise values up to about 0.04 mG and 0.5 μG (see Figure 7.42 and 7.43). The LVM method enables an optimization of the position in the millimeter range from a noise value of about 0.4 μG, as shown in Figure 7.44. The LVM approach shows no effect from a noise value of about 0.01 μG (see Figure 7.45).



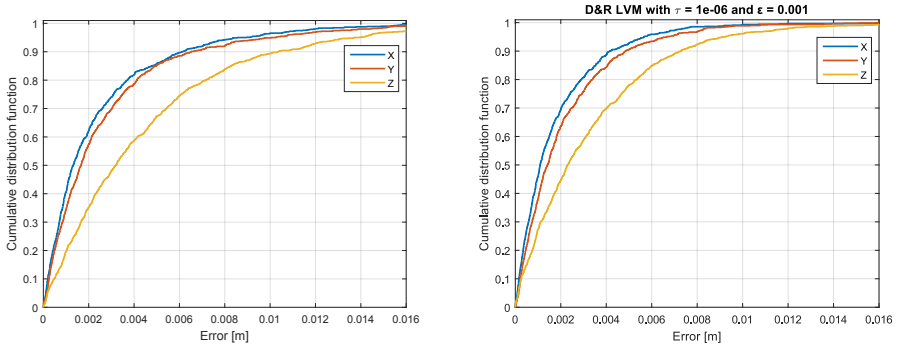
(a) **(b)**

Figure 7.42.: *ML and LVM algorithms for noise equal to 0.04 mG*



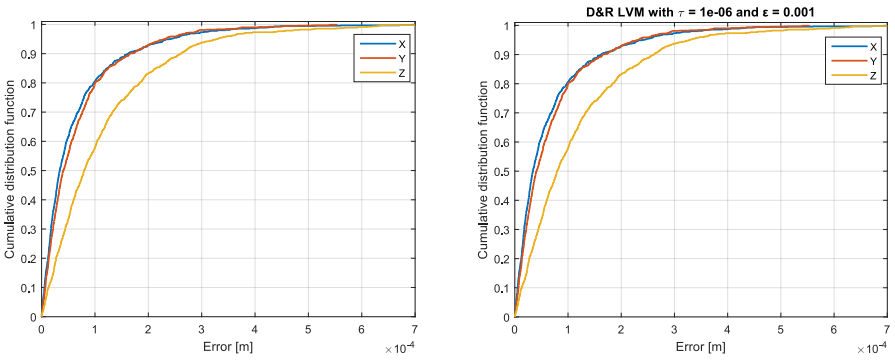
(a) **(b)**

Figure 7.43.: *ML and LVM algorithms for noise equal to 0.5 μG*



(a) **(b)**

Figure 7.44.: *ML and LVM algorithms for noise equal to 0.4 μG*



(a) **(b)**

Figure 7.45.: *Multilateration (ML) and Levenberg–Marquardt (LVM) algorithms for noise equal to* $0.01\,\mu G$. *(**a**) ECDF of the ML algorithm, (**b**) ECDF of the LVM algorithm. ECDF, Empirical Cumulative Distribution Function*

7.7.3. Parameter Analysis

The τ-parameter with the value of 10^{-6} shows almost the smallest mean iteration count for ε-values between 10^{-6} and 10^{-1}, as demonstrated in Figure 7.46.

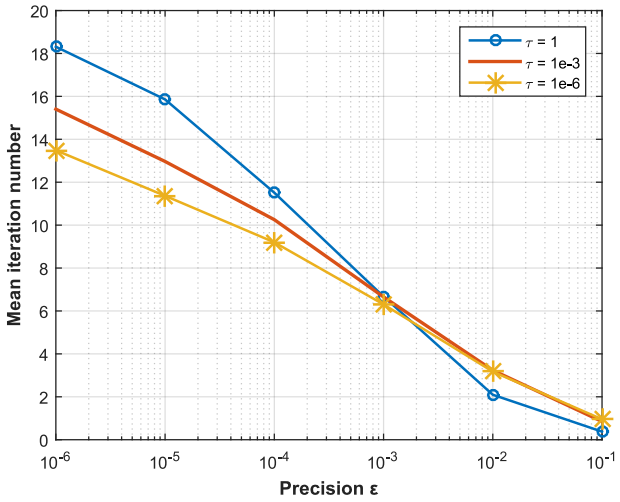


Figure 7.46.: *MILPS: mean iteration numbers of the LVM algorithm for* $\tau = 1, 10^{-3}, 10^{-6}$

The LVM algorithm shows the same and the smallest absolute errors for the parameters $\tau = 10^{-3}$ as well as $\tau = 10^{-6}$ for the values of ε from 10^{-6} up to 10^{-3}. The LVM method shows a slight difference by $\tau = 1$ and $\varepsilon = 10^{-3}$ but shows the same performance for all the τ values for the values of ε between 10^{-6} and 10^{-4}. The tuple $(\tau, \varepsilon) = (10^{-6}, 10^{-3})$ is chosen, since it allows the position optimization with the smallest position error and iteration number. For the sake of simplicity, Figures 7.47 shows only the ECDF of the

multilateration as well as of the LVM algorithm for the parameter $\tau = 10^{-6}$. The LVM method performs with the same level of accuracy for the ε values less than or equal to 10^{-3}, as seen in Figure 7.47. Therefore, the value of ε is chosen equal to 10^{-3} to minimize the absolute position error as well as the iteration numbers of the LVM approach.

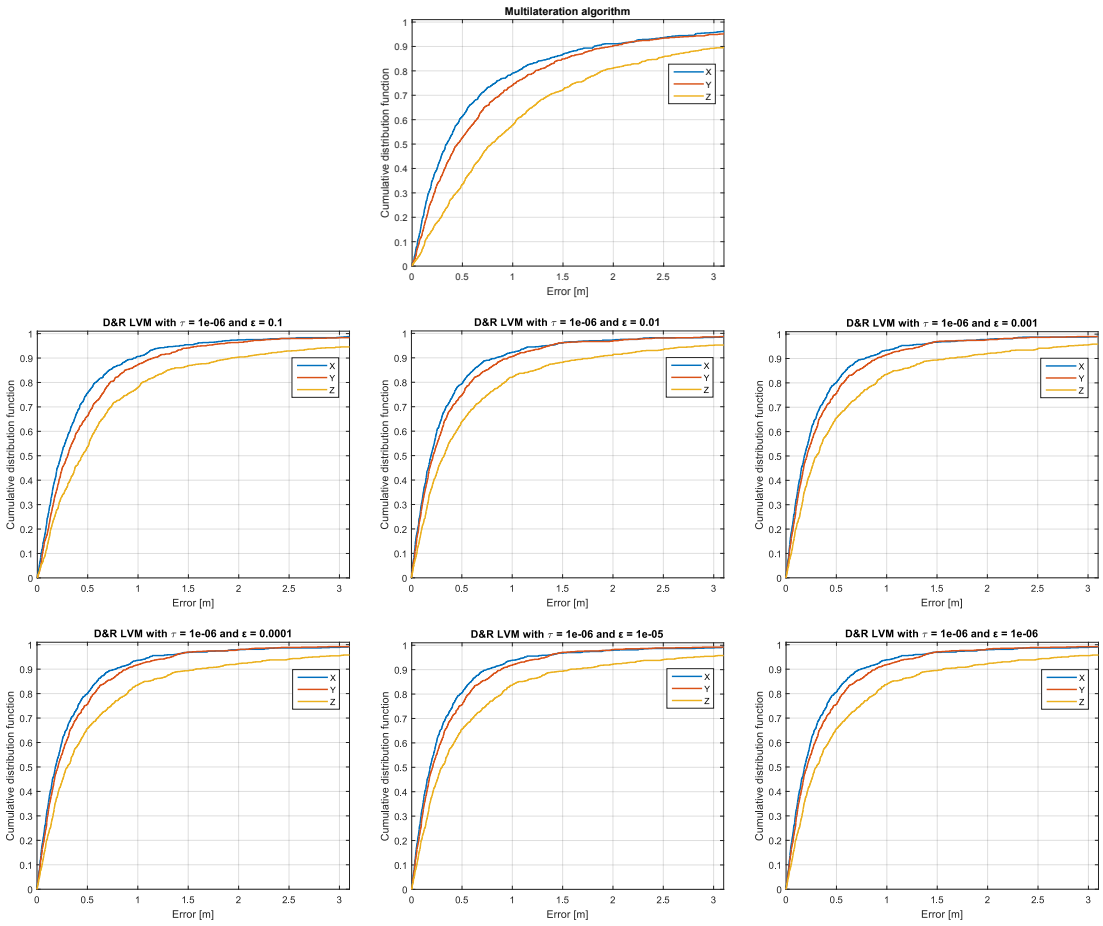


Figure 7.47.: *ECDF of the DR-LVM algorithm with* $\tau = 10^{-6}$

7.8. Conclusion

Cholesky, LU, QR, and SVD algorithms are important matrix decompositions to solve equations for the lateration problem. Cholesky and LU decompositions are unsuitable because of rounding errors, and instability, as well as the limitation to quadratic matrices. We demonstrated that the Householder transformation, which is a method to implement the QR decomposition, is efficient as well as has guaranteed stability. The Householder transformation fails, if the matrix is rank-deficient or nearly rank-deficient. SVD algorithm is the most reliable among the three methods, since it is stable and able to reliably deal with the rank-deficiency or near rank-deficiency of a matrix. In contrast, the SVD method

is (by far) the most expensive.

The solution of the multilateration algorithm by using the SVD is not always optimal, since the accuracy of the searched point is affected by many factors such as its relative position to the reference stations, the coverage (number) of the reference stations, the accuracy of the measured data, and the measurement model. Therefore, the position can be optimized by using NLS methods such as GN or LVM algorithms. Since the GNM can diverge, it is unsuitable for practical implementations. In contrast, the LVM algorithm has a guaranteed convergence, whereby the choice of the right parameter is of utmost importance. The parameter selection impacts the efficiency and the success rate of the LVM algorithm. The parameter choice can be achieved by analyzing the deployment site based on sensors accuracy, PDOP values, the quality of the starting point, and the measurement model. These algorithms enable an accurate and smooth localization determination as well as an adaptive selection of the optimization algorithms locally on the MS. Furthermore, the MS deployed in the UWB-ILS can operate in NLoS environments by using the multipath detection and mitigation (MDDM) algorithm . This algorithm is designed and implemented for devices with limited resources. We implemented as well as evaluated the algorithms in a simulated and in a real environment. We used the resources-constrained devices such as STM32F4 and Raspberry Pi 3 to measure the performance of the algorithms in a real environment.

We demonstrated that there is a correlation between the absolute errors and the PDOP values as well as that the invocation of the LVM method is not always necessary by UWB-ILS. On the contrary, there is no correlation between the absolute errors and the PDOP values by MILPS. Therefore, the use of the LVM algorithm is indispensable. The invocation of the LVM method can be saved on MILPS if the environment, as well as the sensors, are low-noise.

CHAPTER 8

Example Applications

In this chapter, we briefly describe three applications in which the platform is deployed. The first one enables the localization of objects or persons inside a building. The second application enables tracking of moving persons inside a building by overcoming the impact of spatially varying ambient magnetic fields on MILPS. Finally, a robot is tracked in the third application.

8.1. Indoor Localization in a Building

Three coils were placed outside and around the building to achieve indoor localization (see Figure 8.1(a)). This can be achieved up to the third floor, as demonstrated by the magnet strengths captured in the third floor (see Figure 8.1(c)). Figure 8.1(c) shows that the magnet signals could be good detected in the third floor.

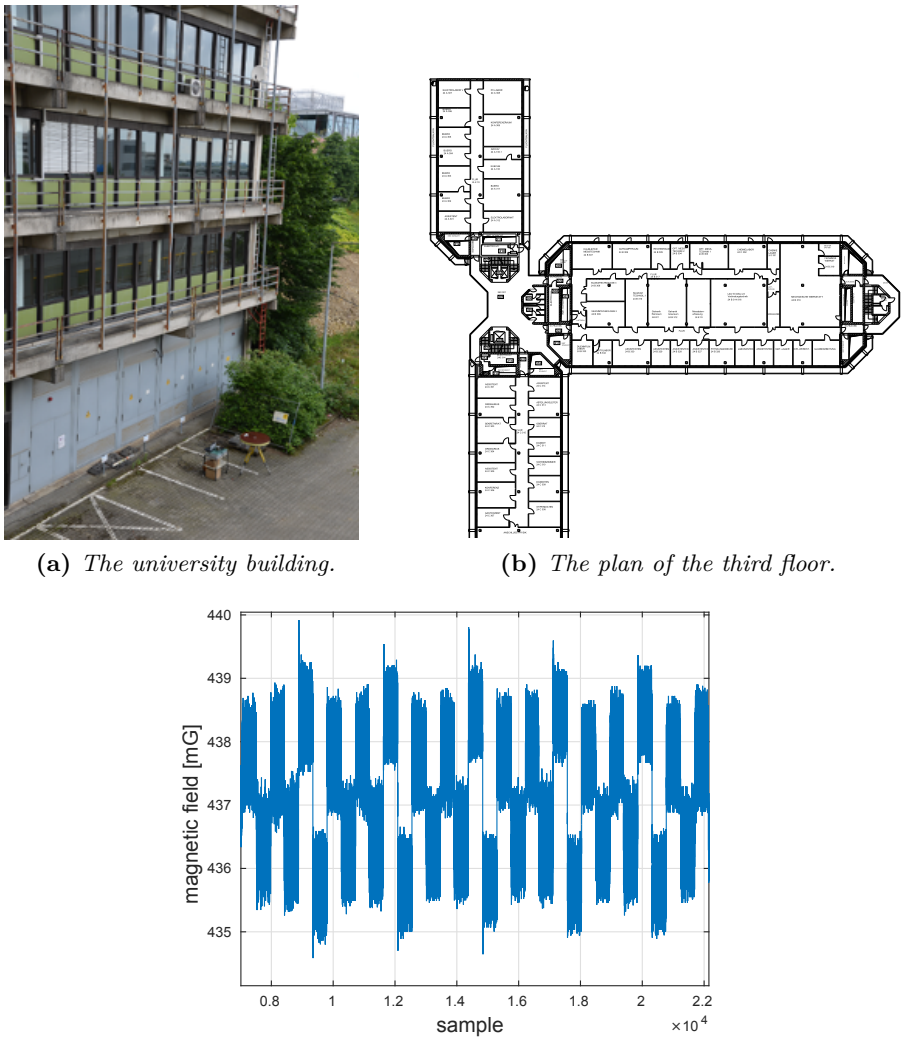
The position accuracy is evaluated by comparing the true and the measured positions by the MILPS. An accuracy of about 50 cm is achieved. Table 8.1 summarizes the relative errors of the measured points at the first and the third floor.

Table 8.1.: *MILPS: the 3-D positions in the first and the third floor*

Position	**Floor**	$\Delta X[m]$	$\Delta Y[m]$	$\Delta Z[m]$
1	1	−0.06	0.15	0.042
2	1	0.46	0.19	−1.11
3	1	−0.17	0.06	1.62
4	3	−0.08	0.11	−0.13
5	3	−0.22	−0.68	0.06

8.2. Foot-Mounted Navigation Using MILPS

The platform architecture is deployed for foot-mounted localization by using MILPS [18]. Due to the advantageous characteristics of magnetic signals such as penetration of various obstacles and robustness against fading effects, MILPS permits positioning in harsh environments. Nevertheless, indoor tracking remains a challenge, due to the non-stationary property of the ambient magnetic fields, signal transients and eddy current effects. These disturbing effects cannot be eliminated during the tracking phase as a result of its spatially-varying character.



(a) *The university building.*

(b) *The plan of the third floor.*

(c) *The magnetic signal strength at the third floor.*

Figure 8.1.: *MILPS deployment in a real-world scenario*

Figure 8.2 shows real captured magnetic clusters in a stationary as well as in the movement phase of a person. In the stationary phase, the ambient magnetic field (offset) is constant and can be mitigated by applying the difference according to (6.1). In the movement stage, the magnetic clusters are distorted because of the varying, overlying ambient magnetic field. The estimation of the coil magnetic field using (6.1) fails in this case. Furthermore, the cluster leading edges are also strongly affected, thus synchronization adjustments would briefly fail. We use MILPS as well as a foot-mounted IMU to overcome the impact of spatially-varying ambient magnetic fields. This method is suitable for pedestrian tracking, since the magnetometer is in a zero-velocity state during the stance phase. The zero-velocity state is detected by measuring the acceleration or the turn rates by means of an IMU sensor [18].

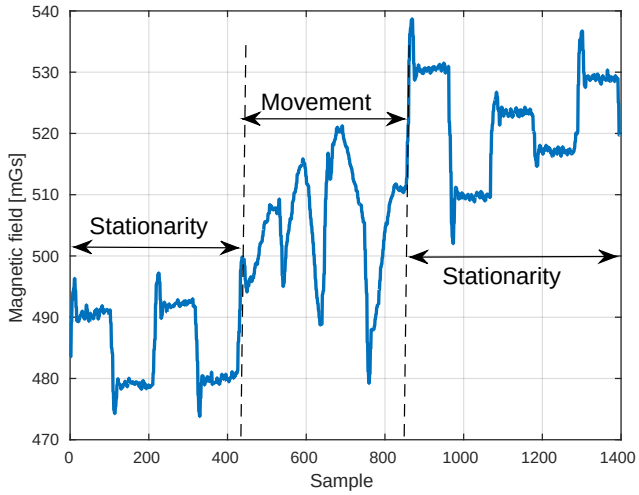


Figure 8.2.: *Captured magnetic field signal clusters: distortion-free clusters (stationary case: left and right part); distorted clusters (movement case: middle part)*

Figure 8.3 shows a real-world indoor environment, where various objects such as metal bookcases, a table, and other furniture are placed. The tracked person is wearing a shoe-mounted IMU to capture the magnetic field and the angular rate data. The person walked along predefined paths using marked points with known coordinates.

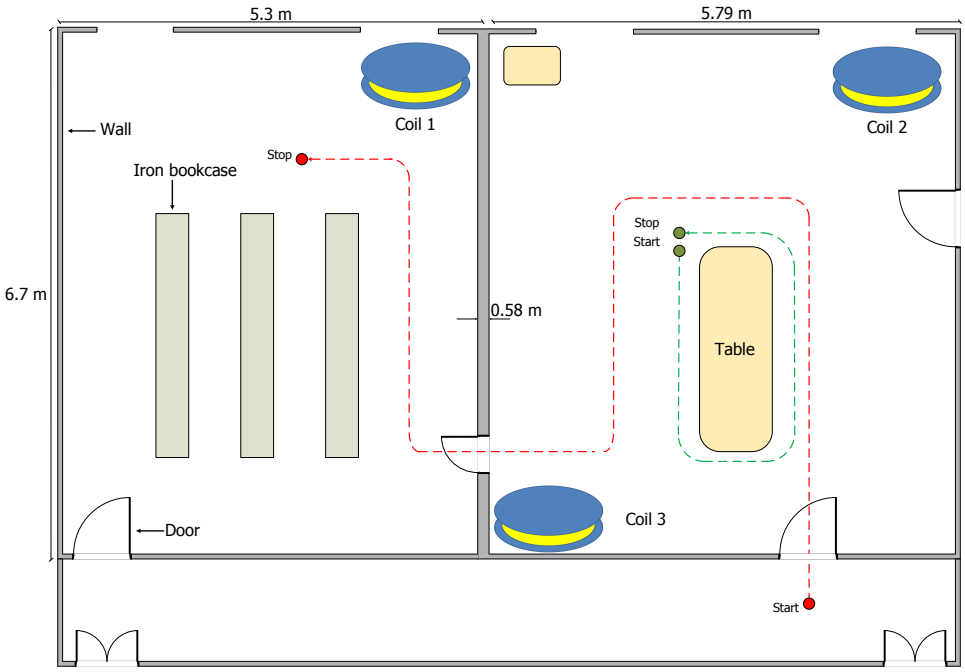


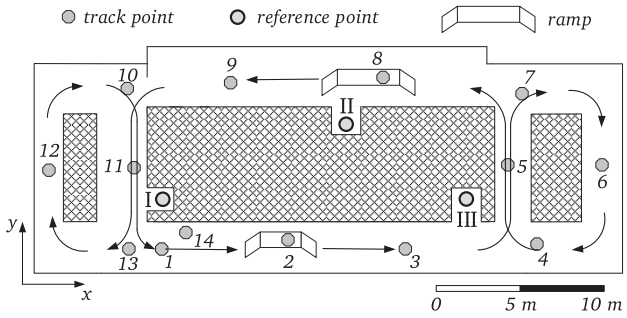
Figure 8.3.: *Experiment setup of foot-mounted navigation by using MILPS and an IMU*

8.3. Robot Localization

The platform architecture is deployed to localize a robot that travels along predefined closed paths on a floor [17]. This platform combines artificially generated magnetic fields with the data of the inertial measurement unit. The IMU is used to cover areas that cannot be reached from the MILPS. The localization system is implemented by using a sensor fusion method as well as a kinematic motion model based on a Kalman Filter. Furthermore, air pressure observations are gathered in combination with an adaptive filtering approach to provide information about altitude changes. Figure 8.4(a) shows the robot traversing a ramp. Figure 8.4(b) illustrates the tracking paths of the mobile robot driven with a velocity of about 1 m/s along various closed loops. One round consists of the following consecutive track points:

1 - 2 - 3 - 5 - 7 - 6 - 4 - 5 - 8 - 9 - 11 - 13 - 12 - 10 - 11 - 1

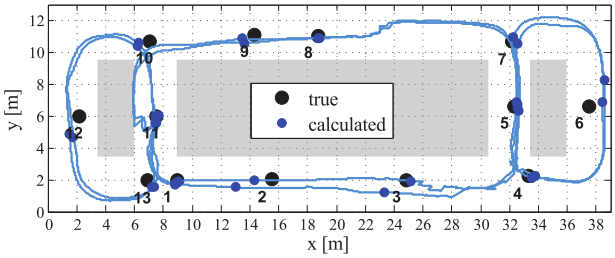
(a) *Robot Platform.*



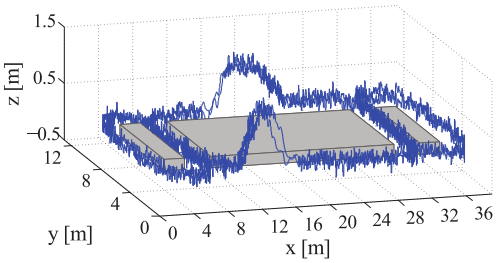
(b) *Tracking path of the robot.*

Figure 8.4.: *Robot tracking with the fusion of MILPS and IMU [17]*

The experimental results show that the robot can be localized with accuracies of less than 1.5 m and 0.5 m in the horizontal plane and in z-direction, respectively (*cf.* Figure 8.5).



(a) *True and estimated positions on the xy plane.*



(b) *Tracking paths in the xyz plane.*

Figure 8.5.: *Tracking results of the robot in two- and three-dimensions [17]*

CHAPTER 9

Summary and Outlook

Localization techniques enable applications such as car navigation systems, emergency services, or tourist tour planning. Furthermore, localization can be a value-added service for other areas such as the automation of industrial systems and equipment, medicine, autonomously acting systems, and networked mobility. To realize such applications, a platform architecture is needed.

The proposed designed platform architecture can be implemented on resource-constrained devices such as microcontrollers, which can build an Internet of thing of networked embedded objects. The combination of real-time localization, Internet, or embedded sensors allows transforming everyday objects into smart objects that can interpret, perceive, and interact with the environment. Therefore, the suggested platform uses the IoT operating system RIOT-OS, which is compatible with various devices such as TI MSP430, AVR Atmega, or ARM Cortex-M3-4. The RIOT-OS enables the realization of non-proprietary and open platform architectures, since it supports the 6LoWPAN, Internet Protocol version 6 (IPv6) and User Datagram Protocol (UDP) facilitating the interoperability with existing systems and protocols. The interoperability is extended by using the JavaScript object notation to enable data exchange with machines. For example, the data exchange between the mobile station and applications located on a PC. JSON is a text-based, lightweight, and language-independent data interchange format. Furthermore, the use of the operating system for resource-constrained devices, which is a software part, provides benefits to applications such as simplifying the design, hiding hardware complexity, and increasing the portability of the source code.

The architecture of the platform is layered- and component-based to enable the use of various localization technologies and positioning algorithms as well as the reuse of software components such as the preprocessing or the matrix operation algorithms. The following points, which are related to the fundamental limitation of preprocessing, matrix computation, and optimization algorithms for positioning by resource-constrained devices, are addressed:

1. A preprocessing method to remove outliers in measured data, which is convenient to resource-constrained devices especially in terms of limited stack memory.
2. A method was introduced, which calculates the 3-D position based on the singular value decomposition.

3. A preprocessing method was proposed for the localization calculation, avoiding the execution of memory and computationally expensive algorithms such as the SVD or the Moore–Penrose pseudo-inverse on resource-constrained devices. Whereby, the complex computation can be shifted to an external device, which possesses more memory and computing capacity. The resource-constrained device is initialized only once with the result of the inverse matrices before deployment.
4. Various methods are described that are suitable to compute an unoptimized position (start position). These algorithms have been analyzed and compared in respect to the stability, complexity and memory requirements.
5. The demonstration of the feasibility to deploy the Moore–Penrose algorithm, which is based on SVD, on resource-constrained devices.
6. The improvement of the position estimate by using the Gauss–Newton as well as the Levenberg–Marquardt algorithms, which are derived in a convenient form for resource-constrained devices.
7. An adaptive algorithm for the optimization of the position was developed, which is based on the SVD, Levenberg–Marquardt algorithm and the position dilution of precision. This algorithm supports an adaptive selection mechanism for the Levenberg–Marquardt algorithm. This adaptive algorithm enables savings in resources such as memory, computing time, and energy on resource-constrained devices. Furthermore, two variants of the LVM algorithms are used: the Dahmen-Reusken LVM and Madsen LVM are analyzed and compared with the Gauss–Newton method. All the algorithms are derived in convenient form for resource-constrained devices. Since the parameters of the LVM algorithm impact the accuracy as well as the required iteration number, the influence and the choice of the right parameter combination have been determined, analyzed and discussed.
8. The energy consumption of the algorithms as well as of the mobile station for the UWB-based ILS and the magnetic indoor local positioning system are evaluated on resource-constrained devices (LPC2387- and STM32F4-based boards).
9. The design and evaluation of a method to reduce multipath errors on the mobile station, which enables an accurate localization in non-line of sight scenarios. This method is implemented as well as evaluated in a simulated and real environment. The STM32F4 microcontroller as well as in Raspberry Pi 3 are used to evaluate the method in real environment.

The platform architecture is tested for UWB and for magnetic field-based technologies. The proposed magnetic-field based system is a stand-alone localization system that enables a positioning in harsh conditions without either the need for communication infrastructure nor a fixed or tedious installation. The main contribution of this system is the proposal of a decentralized control of the individual coils (anchors), as well as the decentralized

synchronization of the entire system without the need for communication technology. Both the synchronization and the control of the coils and MS are based on a preemptive real-time operating system and real-time clocks. Furthermore, the developed work of the magnetic-based system can be summarized as follows:

1. The design of a decentralized positioning system by improving the MILPS using coil driver units based on accurate RTCs. Furthermore, the MS is extended with a sensor platform, which includes a magnetic field sensor and an RTC. The MS operates independently from the CDUs, and no communication channel is required.
2. The application of time division multiple access for the generation of periodic, distortion-free magnetic field signals for a certain time period (e.g., 1 s). The TDMA allows the MS to distinguish between the coils (reference points).
3. The evaluation of two approaches to drive and synchronize the coils.
4. The development, analysis, and evaluation of algorithms for MILPS based on resource-limited devices.

The UWB-based ILS has a maximal error of approximately 3.5 cm in the x- and y-coordinates and a maximal error of approximately 25.3 cm in the z-coordinates by using the Multilateration (ML) algorithm. In this case, the reference points are located at approximately the same altitude to impair the position in the z-direction and to stress the algorithms. The 3D position can be refined by using the GN algorithm that achieves a maximal error of approximately 2.2 cm in the x- and y-coordinates and a maximal error of approximately 11.2 cm in the z-coordinates. The position can also be improved by applying the LVM algorithm that is stable, since it delivers the same results if the GN algorithm does not diverge. Furthermore, the LVM algorithm delivers a stable position, if the GN algorithm diverges and shows a poor performance. In the NLoS scenarios, the position can be optimized by using the MDDM algorithm that can reduce the position errors up to approximately 40 cm as well as the errors in the z-coordinates up to 0.6 m.

MILPS has a maximal error of approximately 30 cm in the x- and y-components by using the ML algorithm. However, the z-component of the MS coordinates shows the poorest performance, if the coils are placed at nearly equal heights. The invocation of the LVM method, which is necessary, can reduce the errors in the x- and y-coordinates to approximately 10 cm. The LVM algorithm can limit the errors in the z-coordinates to approximately 0.6 m. The LVM method has generally lower deviations in all coordinate components compared with other methods such as the ML or the GN method.

Finally, the platform architecture was deployed in various real-world scenarios. The proposed platform architecture can also be a basis for a distributed localization approach. This is a fruitful research area, since the majority of multihop localization techniques are not implemented, are only treated at the theory level or tested in simulated environments.

APPENDIX A

Open Source Software Library for Numerical Linear Algebra and Localization (RcdMathLib)

RcdMathLib is an open-source library for resource-constrained devices such as microcontrollers that provides linear algebra, localization, and optimization algorithms. RcdMathLib has a modular and pyramidal architecture as shown in Figure A.1, whereby each layer rests upon the underlying layers. The RcdMathLib is composed of three main components: linear algebraic, localization, and optimization layers.

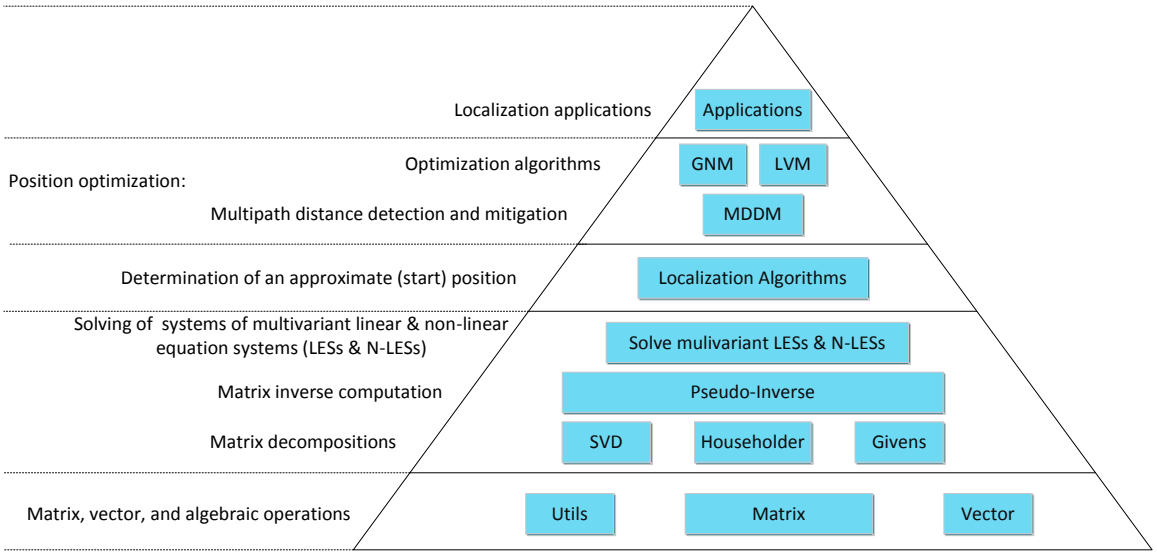


Figure A.1.: *Architecture of the RcdMathLib*

A.1. Linear Algebraic Layer

The linear algebraic layer builds the base of the whole library and provides basic as well as complex linear algebraic operations. Basic algebraic operations are, for example, the

addition or multiplication of vectors or matrices; while complex operations are the calculation of the inverse, pseudo-inverse, or the solution of over-determined linear equation systems. The linear algebraic layer also allows solving multivariant non-linear equations. All these operations are implemented based on efficient algorithms such as QR-Householder or SVD methods, as well as in a convenient form for resource-constrained devices. Furthermore, the util-module provides various algorithms such as the Shell Sort algorithm or the Box-Muller method to generate normally distributed random variables.

A.2. Localization Layer

The localization layer enables the calculation of a position based on algorithms such as trilateration algorithm. The localization algorithms also provide for the measurement as well as the error models of the respective localization technique. They also enable the computation of the Jacobian matrices, which are needed for the optimization layer.

A.3. Optimization Layer

The optimization layer allows the detection and mitigation of multipath effects in the case of distance-based localization technique. This Layer enables also the refinement of a position delivered from the localization layers by using optimization algorithms such as GNM or LVM algorithm.

A.4. Application Layer

Users can implement efficient localization applications such as the tracking of persons or objects by using the previously described layers.

ANHANG B

Zusammenfassung

Eine Plattformarchitektur ist hilfreich für die Realisierung von flexiblen Lokalisierungssystemen, die sowohl mit verschiedenen Systemen interagieren als auch zahlreiche Lokalisierungstechnologien und -Algorithmen unterstützen. Die dezentrale Bearbeitung von einer Position ermöglicht die Verschiebung der Anwendungsschicht in die mobile Station und so vermeidet sie die Kommunikation mit einer zentralen Recheneinheit wie z.B. einem Server oder einer Basisstation. Außerdem stellt die Positionsberechnung in einem kostengünstigen und ressourceneingeschränkten Gerät eine Herausforderung wegen begrenzter Rechen- und Speicherkapazität sowie Energieversorgung dar. Deshalb bietet diese Dissertation eine Plattformarchitektur, die für einen Systemdesigner die folgenden Vorteile ermöglicht: Wiederverwendbarkeit von fertigen Modulen, Erweiterbarkeit z.B. mit anderen Lokalisierungstechnologien, und Interoperabilität. Zeitgleich (on-the-fly) wird die Position in einem kostengünstigen Gerät wie ein Mikrocontroller berechnet. Basierend auf einem Betriebssystem führt das Gerät gleichzeitig zusätzliche Arbeitsschritte wie Datensammlung oder -bearbeitung aus. Die Plattformarchitektur wurde für zwei Systeme entworfen, implementiert und evaluiert: entfernungs- und feldstärkebasierte Positionssysteme. Beide Systeme verwenden jeweils die Ultra-Breitband-Technologie (UWB) und gepulste Gleichstrom-Magnetfelder. Geeignete Algorithmen wurden für die Berechnung einer nicht optimalen Position (Startposition) vorgeschlagen. Diese Algorithmen wurden hinsichtlich Stabilität, Effizienz, Komplexität und Speicherbedarf verglichen und analysiert. Ein adaptiver Algorithmus für die Positionsoptimierung wurde basierend auf der Singulärwertzerlegung (SWZ), dem Levenberg–Marquardt (LVM) Algorithmus und dem Dilution of Precision (DOP)-Wert entwickelt. Der DOP-Wert ist ein Maß über die relative Position der Spulen zueinander und zur gesuchten Position. Der adaptive Algorithmus ermöglicht die Einsparung von Ressourcen wie bespielweise Speicher, Rechenzeit und Stromverbrauch von Geräten mit eingeschränkten Ressourcen. Außerdem wurden zwei Varianten von den LVM-Algorithmen verwendet: die Dahmen-Reusken LVM und Madsen LVM Algorithmen. Beide Algorithmen wurden mit dem Gauss–Newton Algorithmus verglichen und analysiert. Sämtliche Algorithmen wurden in eine geeignete Form für Ressourcen-eingeschränkte Geräte abgeleitet. Die Parameterauswahl vom LVM-Algorithmus beeinflusst sowohl die Genauigkeit als auch die erforderte Iterationszahl. Demzufolge wurde der Einfluss and die Auswahl von der genauen Parameterkombination untersucht, analysiert und diskutiert. Schlussendlich wurde eine Methode zur Reduzierung der Mehrwege-Fehler in mobilen Stationen entwickelt und bewertet. Diese Methode ermöglicht eine präzise Lokalisierung in Szenarien ohne direkte Sichtverbindung zu den Referenzstationen.

ANHANG C

Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Dissertation auf Grundlage der in der Arbeit angegebenen Hilfsmittel und Hilfen selbständig verfasst habe. Die Arbeit ist nicht in einem früheren Promotionsverfahren eingereicht worden.

Berlin,

Bibliography

[1] Brena, R. F.; García-Vázquez, J. P.; Galván-Tejada, C. E.; Muñoz-Rodrigue, D.; Vargas-Rosales, C.; Fangmeyer, J.: *Evolution of Indoor Positioning Technologies: A Survey*. In: *Journal of Sensors* 2017 (2017-03), pp. 1–21. doi: 10.1155/2017/2630413.

[2] Hentschel, M.; Wulf, O.; Wagner, B.: *A GPS and Laser-based Localization for Urban and Non-Urban Outdoor Environments*. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nice, France: IEEE, 2008-09, pp. 149–154. isbn: 978-1-4244-2057-5. doi: 10.1109/IROS.2008.4650585.

[3] Xu, R.; Chen, W.; Xu, Y.; Ji, S.: *A New Indoor Positioning System Architecture Using GPS Signals*. In: *Sensors* 15.5 (2015-04), pp. 10074–10087. issn: 1424-8220. doi: 10.3390/s150510074.

[4] Kanli, M. O.: *Limitations of Pseudolite Systems Using Off-The-Shelf GPS Receivers*. In: *Proceedings of the 2004 International Symposium on GNSS/GPS*. Sydney, Australia, 2004-12, pp. 154–166.

[5] Deak, G.; Curran, K.; Condell, J.: *A Survey of Active and Passive Indoor Localisation Systems*. In: *Computer Communications* 35.16 (2012-09), pp. 1939–1954. issn: 0140-3664. doi: 10.1016/j.comcom.2012.06.004.

[6] Linde, H.: *On Aspects of Indoor Localization*. PhD thesis. Department of Electrical Engineering, University of Dortmund, 2006.

[7] Azizyan, M.; Constandache, I.; Roy Choudhury, R.: *SurroundSense: Mobile Phone Localization via Ambience Fingerprinting*. In: *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*. MobiCom '09. Beijing, China: ACM, 2009-09, pp. 261–272. isbn: 978-1-60558-702-8. doi: 10.1145/1614320.1614350.

[8] Muthukrishnan, K.; Lijding, M.; Havinga, P.: *Towards Smart Surroundings: Enabling Techniques and Technologies for Localization*. In: *International Symposium on Location- and Context-Awareness (LoCa 2005)*. Berlin, Heidelberg, Germany: Springer Berlin Heidelberg, 2005, pp. 350–362. isbn: 978-3-540-32042-5. doi: 10.1007/11426646_32.

[9] Bulusu, N.; Heidemann, J.; Heidemann, J.; Estrin, D.; Tran, T.: *Self-Configuring Localization Systems: Design and Experimental Evaluation*. In: *ACM Transactions on Embedded Computing Systems (TECS)* 3.1 (2004-02), pp. 24–60. issn: 1539-9087. doi: 10.1145/972627.972630.

[10] Liu, H.; Darabi, H.; Banerjee, P.; Liu, J.: *Survey of Wireless Indoor Positioning Techniques and Systems*. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.6 (2007-11), pp. 1067–1080. issn: 1094-6977. doi: 10.1109/TSMCC.2007.905750.

[11] Zekavat, R.; Buehrer, R. M.: *Handbook of Position Location: Theory, Practice and Advances*. IEEE Series on Digital & Mobile Communication. Piscataway, NJ Hoboken, New Jersey, USA: IEEE Press John Wiley & Sons, Inc, 2012. ISBN: 978-0-470-94342-7.

[12] Baccelli, E.; Hahm, O.; Günes, M.; Wählisch, M.; Schmidt, T. C.: *RIOT OS: Towards an OS for the Internet of Things*. In: *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Turin, Italy: IEEE, 2013-04, pp. 79–80. ISBN: 978-1-4799-0055-8. DOI: 10.1109/INFCOMW.2013.6970748.

[13] Kasmi, Z.; Norrdine, A.; Blankenbach, J.: *Towards a Decentralized Magnetic Indoor Positioning System*. In: *Sensors* 15.12 (2015-12), pp. 30319–30339. ISSN: 1424-8220. DOI: 10.3390/s151229799.

[14] Kasmi, Z.; Norrdine, A.; Blankenbach, J.: *Platform Architecture for Decentralized Positioning Systems*. In: *Sensors* 17.5 (2017-04), pp. 1–29. ISSN: 1424-8220. DOI: 10.3390/s17050957.

[15] Kasmi, Z.; Guerchali, N.; Norrdine, A.; Schiller, J. H.: *Algorithms and Position Optimization for a Decentralized Localization Platform Based on Resource-Constrained Devices*. In: *IEEE Transactions on Mobile Computing* 18.08 (2019-08), pp. 1731–1744. ISSN: 1536-1233. DOI: 10.1109/TMC.2018.2868930.

[16] Kasmi, Z.; Norrdine, A.; Sieprath, A.; Blankenbach, J.: *Accurate Distance Measurement Using The TIME DOMAIN ® P410 UWB Radio*. In: *IPIN 2013 - 4th International Conference on Indoor Positioning and Indoor Navigation (IPIN 2013) ; Montbéliard, France, 28.-31. Oct. 2013*. 2013.

[17] Hellmers, H.; Kasmi, Z.; Norrdine, A.; Eichhorn, A.: *Accurate 3D Positioning for a Mobile Platform in Non-Line-of-Sight Scenarios Based on IMU/Magnetometer Sensor Fusion*. In: *Sensors* 18.1 (2018-01), pp. 1–19. ISSN: 1424-8220. DOI: 10.3390/s18010126.

[18] Norrdine, A.; Kasmi, Z.; Blankenbach, J.: *A Novel Method for Overcoming the Impact of Spatially Varying Ambient Magnetic Fields on a DC Magnetic Field-based Tracking System*. In: *Journal of Location Based Services* 10.1 (2016-04), pp. 3–15. DOI: 10.1080/17489725.2016.1170898.

[19] Norrdine, A.; Kasmi, Z.; Blankenbach, J.: *Step Detection for ZUPT-Aided Inertial Pedestrian Navigation System Using Foot-Mounted Permanent Magnet*. In: *IEEE Sensors Journal* 16.17 (2016-09), pp. 6766–6773. ISSN: 1530-437X. DOI: 10.1109/JSEN.2016.2585599.

[20] Norrdine, A.; Kasmi, Z.; Blankenbach, J.: *Position Estimation by Means of a Magnetometer and Inclinometer - an Algorithm Evaluation*. In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN) : 4-7 October 2016, Alcalá de Henares, Madrid, Spain / organized by: University of Alcalá ; technical sponsors: IEEE, GRSS, IEEE Instrumentation & Measurement Society, IEEE Sección España*. 7th International conference on Indoor Positioning for wheeled platforms in NLoS scenarios, Acalá de Henares (Spain), 4 Oct 2016 - 7 Oct 2016. Piscataway, NJ: IEEE, 2016-10-04, p. 3.

[21] *Aistear: the Early Childhood Curriculum Framework*. National Council for Curriculum and Assessment (NCCA). 2009.

[22] Gu, Y.; Lo, A.; Niemegeers, I.: *A Survey of Indoor Positioning Systems for Wireless Personal Networks*. In: *IEEE Communications Surveys & Tutorials* 11.1 (2009-03), pp. 13–32. ISSN: 1553-877X. DOI: 10.1109/SURV.2009.090103.

[23] Hightower, J.; Borriello, G.: *Location Sensing Techniques*. UW CSE 01-07-01. Seattle, WA: University of Washington, Department of Computer Science and Engineering, 2001-07.

[24] Want, R.; Hopper, A.; Falcão, V.; Gibbons, J.: *The Active Badge Location System*. In: *ACM Transactions on Information Systems (TOIS)* 10.1 (1992-01), pp. 91–102. ISSN: 1046-8188. DOI: 10.1145/128756.128759.

[25] Jochen, S.: *Mobile Communications*. Second Edition. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN: 0-321-12381-6.

[26] Varshavsky, A.; de Lara, E.; Hightower, J.; LaMarca, A.; Otsason, V.: *GSM Indoor Localization*. In: *Pervasive and Mobile Computing* 3.6 (2007-12), pp. 698–720. ISSN: 1574-1192. DOI: 10.1016/j.pmcj.2007.07.004.

[27] Kos, T.; Grgic, M.; Sisul, G.: *Mobile User Positioning in GSM/UMTS Cellular Networks*. In: *Proceedings Electronics in Marine (ELMAR 2006)*. Zadar, Croatia: IEEE, 2006-06, pp. 185–188. ISBN: 953-7044-03-3. DOI: 10.1109/ELMAR.2006.329545.

[28] Partridge, K.; Arnstein, L.; Borriello, G.; Whitted, T.: *Fast Intrabody Signaling*. Demonstration at Wireless and Mobile Computer Systems and Applications (WMCSA). Monterey, CA, USA, 2000-12.

[29] Ghavami, M.; Michael, L.; Kohno, R.: *Ultra Wideband Signals and Systems in Communication Engineering*. Second Edition. Hoboken, New Jersey, USA: John Wiley & Sons, Ltd, 2007. ISBN: 978-0-470-02763-9. DOI: 10.1002/9780470060490.

[30] Hightower, J.: *The Location Stack*. PhD thesis. Seattle, WA: Department of Computer Science & Engineering, University of Washington, 2004.

[31] Al-Ammar, M. A.; Alhadhrami, S.; Al-Salman, A.; Alarifi, A.; Al-Khalifa, H. S.; Alnafessah, A.; Alsaleh, M.: *Comparative Survey of Indoor Positioning Technologies, Techniques, and Algorithms*. In: *2014 International Conference on Cyberworlds*. Santander, Spain: IEEE, 2014-10, pp. 245–252. DOI: 10.1109/CW.2014.41.

[32] Norrdine, A.: *Präzise Positionierung und Orientierung innerhalb von Gebäuden*. Vol. 29. Schriftenreihe Fachrichtung Geodäsie Fachbereich Bauingenieurwesen und Geodäsie. Darmstadt, Germany: Technische Universität Darmstadt; Geodätisches Institut, 2009. ISBN: 978-3-935631-18-1. URL: https://tuprints.ulb.tu-darmstadt.de/6911 (visited on 2019-07-15).

[33] Karl, H.; Willig, A.: *Protocols and Architectures for Wireless Sensor Networks*. Chichester, United Kingdom: John Wiley & Sons, 2005. ISBN: 978-0-470-09510-2. DOI: 10.1002/0470095121.

[34] Saleh, A. A. M.; Valenzuela, R.: *A Statistical Model for Indoor Multipath Propagation*. In: *IEEE Journal on Selected Areas in Communications* 5.2 (1987-02), pp. 128–137. ISSN: 0733-8716. DOI: 10.1109/JSAC.1987.1146527.

[35] Seidel, S. Y.; Rappaport, T. S.: *914 MHz Path Loss Prediction Models for Indoor Wireless Communications in Multifloored Buildings*. In: *IEEE Transactions on Antennas and Propagation* 40.2 (1992-02), pp. 207–217. ISSN: 0018-926X. DOI: 10.1109/8.127405.

[36] Weinstock, H. C.: *Focus on Cognitive Radio Technology*. New York, NY, USA: Nova Science Publishers Inc, 2007. ISBN: 978-1-60021-215-4.

[37] Hakyong Kim: *A Ranging Scheme for Asynchronous Location Positioning Systems*. In: *2009 6th Workshop on Positioning, Navigation and Communication*. Hannover, Germany: IEEE, 2009-03, pp. 89–94. ISBN: 978-1-4244-3292-9. DOI: 10.1109/WPNC.2009.4907809.

[38] Kim, H.: *Double-Sided Two-Way Ranging Algorithm to Reduce Ranging Time*. In: *IEEE Communications Letters* 13.7 (2009-07), pp. 486–488. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2009.090093.

[39] McCrady, D. D.; Doyle, L.; Forstrom, H.; Dempsey, T.; Martorana, M.: *Mobile Ranging Using Low-Accuracy Clocks*. In: *IEEE Transactions on Microwave Theory and Techniques* 48.6 (2000-06), pp. 951–958. ISSN: 0018-9480. DOI: 10.1109/22.846721.

[40] Sahinoglu, Z.; Gezici, S.: *Ranging in the IEEE 802.15.4a Standard*. In: *2006 IEEE Annual Wireless and Microwave Technology Conference*. Clearwater Beach, FL, USA: IEEE, 2006-12, pp. 1–5. ISBN: 1-4244-0848-2. DOI: 10.1109/WAMICON.2006.351897.

[41] Laaraiedh, M.; Avrillon, S.; Uguen, B.: *Cramer–Rao Lower Bounds for Nonhybrid and Hybrid Localisation Techniques in Wireless Networks*. In: *Transactions on Emerging Telecommunications Technologies* 23.3 (2012), pp. 268–280. ISSN: 2161-3915. DOI: 10.1002/ett.1530.

[42] Venkatraman, S.; Caffery, J.: *Hybrid TOA/AOA Techniques for Mobile Location in Non-Line-of-Sight Environments*. In: *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733)*. Vol. 1. Atlanta, GA, USA: IEEE, 2004-03, pp. 274–278. ISBN: 0-7803-8344-3. DOI: 10.1109/WCNC.2004.1311556.

[43] Chen, C.-S.; Chiu, Y.-J.; Lin, J.-M.: *Hybrid TOA/AOA Schemes for Mobile Location In Cellular Communications Systems*. In: *International Journal of Ad Hoc, Sensor & Ubiquitous Computing (IJASUC)* 1.2 (2010-06), pp. 54–64.

[44] Chen, T.-Y.; Chiu, C.-C.; Tu, T.-C.: *Mixing and Combining with AOA and TOA for the Enhanced Accuracy of Mobile Location*. In: *2003 5th European Personal Mobile Communications Conference (Conf. Publ. No. 492)*. Glasgow, UK: IET, 2003-04, pp. 276–280. ISBN: 0-85296-753-5. DOI: 10.1049/cp:20030261.

[45] Randell, C.; Djiallis, C.; Muller, H.: *Personal Position Measurement Using Dead Reckoning*. In: *Proceedings of the 7th IEEE International Symposium on Wearable Computers*. ISWC '03. White Plains, NY, USA: IEEE Computer Society, 2003-10, pp. 166–173. ISBN: 0-7695-2034-0. DOI: 10.1109/ISWC.2003.1241408.

[46] Rantakokko, J.; Händel, P.; Fredholm, M.; Marsten-Eklöf, F.: *User Requirements for Localization and Tracking Technology: A Survey of Mission-specific Needs and Constraints*. In: *2010 International Conference on Indoor Positioning and Indoor Navigation*. IEEE. Zurich, Switzerland, 2010, pp. 1–9. ISBN: 978-1-4244-5862-2. DOI: 10.1109/IPIN.2010.5646765.

[47] Zahid, F.; Rosdiadee, N.; Mahamod, I.: *Recent Advances in Wireless Indoor Localization Techniques and System*. In: *Journal of Computer Networks and Communications* 2013 (2013-08), pp. 1–12. doi: 10.1155/2013/185138.

[48] Ho, S.-S.; Ruan, S.: *Differential Privacy for Location Pattern Mining*. In: *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*. SPRINGL '11. Chicago, Illinois: ACM, 2011, pp. 17–24. isbn: 978-1-4503-1032-1. doi: 10.1145/2071880.2071884.

[49] Mautz, R.: *Indoor Positioning Technologies*. ETH Zurich, Department of Civil, Environmental, Geomatic Engineering, Institute of Geodesy, and Photogrammetry, 2012. isbn: 978-3-908440-31-4.

[50] Werner, M.; Kessel, M.; Marouane, C.: *Indoor Positioning Using Smartphone Camera*. In: *2011 International Conference on Indoor Positioning and Indoor Navigation*. Guimaraes, Portugal: IEEE, 2011-09, pp. 1–6. isbn: 978-1-4577-1805-2. doi: 10.1109/IPIN.2011.6071954.

[51] Zhao, Y.-g.; Cheng, W.; Jia, L.; Ma, S.-l.: *The Obstacle Avoidance and Navigation Based on Stereo Vision for Mobile Robot*. In: *2010 International Conference on Optoelectronics and Image Processing*. Vol. 2. Haikou, China: IEEE, 2010-11, pp. 565–568. isbn: 978-1-4244-8683-0. doi: 10.1109/ICOIP.2010.14.

[52] Desouza, G. N.; Kak, A. C.: *Vision for Mobile Robot Navigation: A Survey*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.2 (2002-02), pp. 237–267. issn: 0162-8828. doi: 10.1109/34.982903.

[53] Mahfouz, M. R.; Fathy, A. E.; Kuhn, M. J.; Wang, Y.: *Recent Trends and Advances in UWB Positioning*. In: *2009 IEEE MTT-S International Microwave Workshop on Wireless Sensing, Local Positioning, and RFID*. Cavtat, Croatia: IEEE, 2009-09, pp. 1–4. isbn: 978-1-4244-5060-2. doi: 10.1109/IMWS2.2009.5307895.

[54] Hol, J. D.; Dijkstra, F.; Luinge, H.; Schon, T. B.: *Tightly Coupled UWB/IMU Pose Estimation*. In: *2009 IEEE International Conference on Ultra-Wideband*. Vancouver, BC, Canada: IEEE, 2009-09, pp. 688–692. isbn: 978-1-4244-2930-1. doi: 10.1109/ICUWB.2009.5288724.

[55] Yamaguchi, H.; Higuchi, T.; Higashino, T.: *Collaborative Indoor Localization of Mobile Nodes*. In: *The Sixth International Conference on Mobile Computing and Ubiquitous Networking (ICMU2012)*. Okinawa, Japan, 2012-05, pp. 156–63.

[56] Kivimäki, T.; Vuorela, T.; Peltola, P.; Vanhala, J.: *A Review on Device-Free Passive Indoor Positioning Methods*. In: *International Journal of Smart Home* 8.1 (2014), pp. 71–94. issn: 1975-4094.

[57] Reed, J. H.: *An Introduction to Ultra Wideband Communication Systems*. First Edition. Upper Saddle River, NJ, USA: Prentice Hall Press, 2005. isbn: 0-13-148103-7.

[58] *Ultra Wideband Ranging and Communication Module*. TIME DOMAIN®. 2013. url: https://www.timedomain.com (visited on 2016-04-25).

[59] *AeroScout Corporation*. url: http://www.aeroscout.com (visited on 2016-04-25).

[60] Holm, S.: *Ultrasound Positioning Based on Time-of-Flight and Signal Strength*. In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. Sydney, NSW, Australia, 2012, pp. 1–6. ISBN: 978-1-4673-1955-3. DOI: 10.1109/IPIN.2012.6418728.

[61] Schneegans, S.; Vorst, P.; Zell, A.: *Using RFID Snapshots for Mobile Robot Self-Localization*. In: *Proceedings of the 3rd European Conference on Mobile Robots (ECMR 2007)*. Freiburg, Germany, 2007-09, pp. 241–246.

[62] Chawathe, S. S.: *Low-Latency Indoor Localization Using Bluetooth Beacons*. In: *2009 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE. St. Louis, MO, USA, 2009-10, pp. 1–7. ISBN: 978-1-4244-5519-5. DOI: 10.1109/ITSC.2009.5309711.

[63] Krumm, J.; Harris, S.; Meyers, B.; Brumitt, B.; Hale, M.; Shafer, S.: *Multi-Camera Multi-Person Tracking for EasyLiving*. In: *Third IEEE International Workshop on Visual Surveillance*. IEEE. Dublin, Ireland, 2000, pp. 3–10. ISBN: 0-7695-0698-4. DOI: 10.1109/VS.2000.856852.

[64] Sahinoglu, Z.; Gezici, S.; Güvenc, I.: *Ultra-Wideband Positioning Systems - Theoretical Limits, Ranging Algorithms, and Protocols*. Cambridge: Cambridge University Press, 2008. ISBN: 978-0-521-87309-3.

[65] Nekoogar, F.: *Ultra-Wideband Communications: Fundamentals and Applications*. First. Prentice Hall Communications Engineering and Emerging Technologies Series. Upper Saddle River, NJ, USA: Prentice Hall Press, 2005. ISBN: 0-13-146326-8.

[66] Mahfouz, M. R.; Kuhn, M. J.; To, G.; Fathy, A. E.: *Integration of UWB and Wireless Pressure Mapping in Surgical Navigation*. In: *IEEE Transactions on Microwave Theory and Techniques* 57.10 (2009-10), pp. 2550–2564. ISSN: 0018-9480. DOI: 10.1109/TMTT.2009.2029721.

[67] Salman, R.; Willms, I.; Sakamoto, T.; Sato, T.; Yarovoy, A.: *Environmental Imaging with a Mobile UWB Security Robot for Indoor Localisation and Positioning Applications*. In: *2013 European Microwave Conference*. Nuremberg, Germany: IEEE, 2013-10, pp. 1643–1646. DOI: 10.23919/EuMC.2013.6686989.

[68] Fall, B.; Elbahhar, F.; Heddebaut, M.; Rivenq, A.: *Time-Reversal UWB Positioning Beacon for Railway Application*. In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Sydney, NSW, Australia: IEEE, 2012-11, pp. 1–8. ISBN: 978-1-4673-1955-3. DOI: 10.1109/IPIN.2012.6418884.

[69] *Zebra Technologies Corporation, Dart Ultra Wideband (UWB) Technology*. URL: https://www.zebra.com/gb/en.html (visited on 2019-07-15).

[70] *Time Domain Corporation, Precision Ranging & Location, Radar Sensing, Communications*. Time Domain is now a part of Humatics. URL: https://www.timedomain.com (visited on 2016-04-25).

[71] *Data Sheet / User Guide PulsON 440*. Rev. 3200317C. TIME DOMAIN®. 2016-03. URL: http://www.timedomain.com (visited on 2016-04-25).

[72] *RTL-Service*. 2019. URL: https://rtlservice.com/en (visited on 2019-07-15).

[73] *DecaWave Company*. URL: https://www.decawave.com/ (visited on 2019-07-15).

[74] Haverinen, J.; Kemppainen, A.: *A Global Self-localization Technique Utilizing Local Anomalies of the Ambient Magnetic Field*. In: *2009 IEEE International Conference on Robotics and Automation*. Kobe, Japan: IEEE, 2009-05, pp. 3142–3147. ISBN: 978-1-4244-2788-8. DOI: 10.1109/ROBOT.2009.5152885.

[75] Song, S.; Hu, C.; Li, M.; Yang, W.; Meng, M.-H.: *Real Time Algorithm for Magnet's Localization in Capsule Endoscope*. In: *2009 IEEE International Conference on Automation and Logistics*. Shenyang, China, 2009-08, pp. 2030–2035. ISBN: 978-1-4244-4794-7. DOI: 10.1109/ICAL.2009.5262602.

[76] Pham, D. M.; Aziz, S. M.: *A Real-Time Localization System for an Endoscopic Capsule Using Magnetic Sensors*. In: *Sensors* 14.11 (2014), pp. 20910–20929. ISSN: 1424-8220. DOI: 10.3390/s141120910.

[77] Asimov, I.: *Asimov's Biographical Encyclopedia of Science and Technology: the lives and achievements of 1195 great scientists from ancient times to the present, chronologically arranged / by Isaac Asimov*. Second Revised Edition. Doubleday Garden City, N.Y, USA, 1982. ISBN: 0-385-17771-2.

[78] Prigge, E.: *A Positioning System with No Line-of-Sight Restrictions for Cluttered Environments*. PhD thesis. Stanford University, CA, USA, 2004.

[79] Serway, R.; Jewett, J.: *Physics for Scientists and Engineers with Modern Physics*. Ninth Edition. Cengage Learning, 2013. ISBN: 978-1-133-95405-7.

[80] Turville, C.; Vaile, B.: *Quicksmart Introductory Physics*. Quicksmart University Guides Series. Pascal Press, 1995. ISBN: 978-1-86441-019-8.

[81] *Ascension Corporation*. URL: https://www.ascension-tech.com (visited on 2019-07-15).

[82] *Polhemus Corporation*. URL: https://polhemus.com (visited on 2019-07-15).

[83] Sheinker, A.; Ginzburg, B.; Salomonski, N.; Frumkis, L.; Kaplan, B.-Z.: *Localization in 3-D Using Beacons of Low Frequency Magnetic Field*. In: *IEEE Transactions on Instrumentation and Measurement* 62.12 (2013-12), pp. 3194–3201. ISSN: 0018-9456. DOI: 10.1109/TIM.2013.2270919.

[84] De Angelis, G.; Pasku, V.; De Angelis, A.; Dionigi, M.; Mongiardo, M.; Moschitta, A.; Carbone, P.: *An Indoor AC Magnetic Positioning System*. In: *IEEE Transactions on Instrumentation and Measurement* 64.5 (2015-05), pp. 1275–1283. ISSN: 0018-9456. DOI: 10.1109/TIM.2014.2381353.

[85] De Angelis, G.; De Angelis, A.; Dionigi, M.; Mongiardo, M.; Moschitta, A.; Carbone, P.: *An Accurate Indoor Position-Measurement System Using Mutually Coupled Resonating Circuits*. In: *2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*. Montevideo, Uruguay: IEEE, 2014-05, pp. 844–849. DOI: 10.1109/I2MTC.2014.6860862.

[86] De Angelis, G.; De Angelis, A.; Pasku, V.; Moschitta, A.; Carbone, P.: *A Hybrid Outdoor/Indoor Positioning System for IoT Applications*. In: *2015 IEEE International Symposium on Systems Engineering (ISSE)*. Rome, Italy: IEEE, 2015-09, pp. 1–6. DOI: 10.1109/SysEng.2015.7302503.

[87] Almuzaini, K. K.; Gulliver, T. A.: *Range-Based Localization in Wireless Networks Using Density-Based Outlier Detection*. In: *Wireless Sensor Network* 2.11 (2010), pp. 807–814. doi: 10.4236/wsn.2010.211097.

[88] Liu, J.; Priyantha, B.; Zhao, F.; Liang, C.-J. M.; Wang, Q.; James, S.: *Towards Discovering Data Center Genome Using Sensor Nets*. In: *Proceedings of the 5th Workshop on Embedded Networked Sensors (HotEmNets)*. Vol. 167. Charlottesville, VA, USA, 2008-06.

[89] Werner-Allen, G.; Lorincz, K.; Johnson, J.; Lees, J.; Welsh, M.: *Fidelity and Yield in a Volcano Monitoring Sensor Network*. In: *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*. OSDI '06. Seattle, Washington: USENIX Association, 2006, pp. 381–396. isbn: 1-931971-47-1.

[90] Kim, S.; Pakzad, S.; Culler, D.; Demmel, J.; Fenves, G.; Glaser, S.; Turon, M.: *Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks*. In: *2007 6th International Symposium on Information Processing in Sensor Networks*. Cambridge, MA, USA: IEEE, 2007-04, pp. 254–263. isbn: 978-1-59593-638-7. doi: 10.1109/IPSN.2007.4379685.

[91] Levis, P.; Gay, D.: *TinyOS Programming*. First Edition. New York, NY, USA: Cambridge University Press, 2009. isbn: 978-0-521-89606-1.

[92] Mainetti, L.; Patrono, L.; Vilei, A.: *Evolution of Wireless Sensor Networks Towards the Internet of Things: A Survey*. In: *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*. Split, Croatia: IEEE, 2011-09, pp. 1–6. isbn: 978-1-4577-1439-9.

[93] Vilajosana, X.; Tuset, P.; Watteyne, T.; Pister, K.: *OpenMote: Open-Source Prototyping Platform for the Industrial IoT*. In: *Ad Hoc Networks: 7th International Conference, AdHocHets 2015, San Remo, Italy, September 1-2, 2015. Proceedings*. Ed. by Mitton, N.; Kantarci, M. E.; Gallais, A.; Papavassiliou, S. Cham: Springer International Publishing, 2015, pp. 211–222. isbn: 978-3-319-25067-0. doi: 10.1007/978-3-319-25067-0_17.

[94] Rashvand, H. F.; Alcaraz Calero, J. M.: *Smart Sensing Architectures*. In: *Distributed Sensor Systems: Practice and Applications*. Chichester, West Sussex, UK: John Wiley & Sons, Ltd, 2012. Chap. 3, pp. 59–113. isbn: 978-0-470-66124-6.

[95] *Atmel AVR2016: RZRAVEN Hardware User's Guide*. Rev.: 8117E-AVR-07/12. Microchip Technology Inc. Chandler, Arizona, USA, 2012. url: http://ww1.microchip.com/downloads/en/appnotes/doc8117.pdf (visited on 2019-07-15).

[96] Penella, M. T.; Albesa, J.; Gasulla, M.: *Powering Wireless Sensor Nodes: Primary Batteries versus Energy Harvesting*. In: *2009 IEEE Instrumentation and Measurement Technology Conference*. Singapore, Singapore: IEEE, 2009-05, pp. 1625–1630. isbn: 978-1-4244-3352-0. doi: 10.1109/IMTC.2009.5168715.

[97] Vermesan, O.; Friess, P.; Guillemin, P.; Gusmeroli, S.; Sundmaeker, H.; Bassi, A.; Jubert, I. S.; Mazura, M.; Harrison, M.; Eisenhauer, M., et al.: *Internet of Things Strategic Research Roadmap*. In: *Internet of Things: Global Technological and Societal Trends* 1 (2011), pp. 9–52.

[98] Walls, C.: *Selecting an Operating System for Embedded Applications*. In: *Mentor Graphics* (2014).

[99] Farooq, M. O.; Kunz, T.: *Operating Systems for Wireless Sensor Networks: A Survey*. In: *Sensors* 11.6 (2011-05), pp. 5900–5930. ISSN: 1424-8220. DOI: 10.3390/s110605900.

[100] *FreeRTOS Operating System Organisation*. URL: https://www.freertos.org (visited on 2019-07-15).

[101] Levis, P.; Madden, S.; Polastre, J.; Szewczyk, R.; Whitehouse, K.; Woo, A.; Gay, D.; Hill, J.; Welsh, M.; Brewer, E.; Culler, D.: *TinyOS: An Operating System for Sensor Networks*. In: *Ambient Intelligence*. Springer Verlag Heidelberg, 2005, pp. 115–148. ISBN: 978-3-540-23867-6. DOI: 10.1007/3-540-27139-2_7.

[102] Dunkels, A.; Gronvall, B.; Voigt, T.: *Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors*. In: *29th Annual IEEE International Conference on Local Computer Networks*. Tampa, Florida, USA: IEEE, 2004-11, pp. 455–462. ISBN: 0-7695-2260-2. DOI: 10.1109/LCN.2004.38.

[103] Bhatti, S.; Carlson, J.; Dai, H.; Deng, J.; Rose, J.; Sheth, A.; Shucker, B.; Gruenwald, C.; Torgerson, A.; Han, R.: *MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms*. In: *Mobile Networks and Applications* 10.4 (2005-08), pp. 563–579. ISSN: 1383-469X. DOI: 10.1007/s11036-005-1567-8.

[104] Eswaran, A.; Rowe, A.; Rajkumar, R.: *Nano-RK: an Energy-aware Resource-centric RTOS for Sensor Networks*. In: *26th IEEE International Real-Time Systems Symposium (RTSS'05)*. Miami, FL, USA: IEEE, 2005-12, pp. 256–265. ISBN: 0-7695-2490-7. DOI: 10.1109/RTSS.2005.30.

[105] Cao, Q.; Abdelzaher, T.; Stankovic, J.; He, T.: *The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks*. In: *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*. IPSN '08. St. Louis, MO, USA: IEEE Computer Society, 2008, pp. 233–244. ISBN: 978-0-7695-3157-1. DOI: 10.1109/IPSN.2008.54.

[106] Déharbe, D.; Galvão, S.; Moreira, A. M.: *Formalizing FreeRTOS: First Steps*. In: *Formal Methods: Foundations and Applications: 12th Brazilian Symposium on Formal Methods, SBMF 2009 Gramado, Brazil, August 19-21, 2009 Revised Selected Papers*. Ed. by Oliveira, M. V. M.; Woodcock, J. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 101–117. ISBN: 978-3-642-10452-7. DOI: 10.1007/978-3-642-10452-7_8.

[107] Will, H.; Schleiser, K.; Schiller, J.: *A Real-Time Kernel for Wireless Sensor Networks Employed in Rescue Scenarios*. In: *2009 IEEE 34th Conference on Local Computer Networks*. Zurich, Switzerland, 2009-10, pp. 834–841. ISBN: 978-1-4244-4488-5. DOI: 10.1109/LCN.2009.5355049.

[108] Simonović, M.; Saranovac, L.: *Power Management Implementation in FreeRTOS on LM3S3748*. In: *Serbian Journal of Electrical Engineering* 10.1 (2013-02), pp. 199–208. DOI: 10.2298/SJEE1301199S.

[109] Gauger, M.: *Integration of Wireless Sensor Networks in Pervasive Computing Scenarios*. Logos Verlag Berlin GmbH, 2010. ISBN: 978-3-8325-2469-2.

[110] Dong, W.; Chen, C.; Liu, X.; Bu, J.: *Providing OS Support for Wireless Sensor Networks: Challenges and Approaches*. In: *IEEE Communications Surveys & Tutorials* 12.4 (2010-05), pp. 519–530. ISSN: 1553-877X. DOI: 10.1109/SURV.2010.032610.00045.

[111] Dunkels, A.; Schmidt, O.; Voigt, T.; Ali, M.: *Protothreads: Simplifying Event-driven Programming of Memory-constrained Embedded Systems*. In: *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. SenSys '06. Boulder, Colorado, USA: ACM, 2006, pp. 29–42. ISBN: 1-59593-343-3. DOI: 10.1145/1182807.1182811.

[112] Oliver, R. S.; Shcherbakov, I.; Fohler, G.: *An Operating System Abstraction Layer for Portable Applications in Wireless Sensor Networks*. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC '10. Sierre, Switzerland: ACM, 2010, pp. 742–748. ISBN: 978-1-60558-639-7. DOI: 10.1145/1774088.1774243.

[113] Masmano, M.; Ripoll, I.; Crespo, A.: *An Overview of the XtratuM Nanokernel*. In: *Proceedings of the Workshop on Operating System Platforms for Embedded Real-Time Applications (OSPERT 2005)*. Palma de Mallorca, Spain, 2005-07.

[114] Yao, J.; Balaei, A. T.; Hassan, M.; Alam, N.; Dempster, A. G.: *Improving Cooperative Positioning for Vehicular Networks*. In: *IEEE Transactions on Vehicular Technology* 60.6 (2011-07), pp. 2810–2823. ISSN: 0018-9545. DOI: 10.1109/TVT.2011.2158616.

[115] Zhou, Z.; Cui, J.-H.; Zhou, S.: *Localization for Large-Scale Underwater Sensor Networks*. In: *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*. Ed. by Akyildiz, I.; Sivakumar, R.; Ekici, E.; Oliveira, J.; McNair, J. Vol. 4479. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 108–119. ISBN: 978-3-540-72606-7. DOI: 10.1007/978-3-540-72606-7_10.

[116] Han, G.; Xu, H.; Duong, T. Q.; Jiang, J.; Hara, T.: *Localization Algorithms of Wireless Sensor Networks: a Survey*. In: *Telecommunication Systems* 52.4 (2013-04), pp. 2419–2436. ISSN: 1572-9451. DOI: 10.1007/s11235-011-9564-7.

[117] Eswaran, S.; Johnson, M.; Misra, A.; Porta, T.: *Adaptive In-Network Processing for Bandwidth and Energy Constrained Mission-Oriented Multi-hop Wireless Networks*. In: *Proceedings of the 5th IEEE International Conference on Distributed Computing in Sensor Systems*. DCOSS '09. Marina del Rey, CA, USA: Springer-Verlag, 2009, pp. 87–102. ISBN: 978-3-642-02085-8. DOI: 10.1007/978-3-642-02085-8_7.

[118] *Ubisense: the Ubisense Precise Real-time Location System (Dimension 4)*. URL: https://www.ubisense.net/product/dimension4 (visited on 2019-07-15).

[119] *The Ekahau Real Time Location System*. Ekahau Inc. URL: https://www.ekahau.com (visited on 2019-07-15).

[120] Wang, W.; Xia, W.; Zhang, R.; Shen, L.: *Design and Implementation of Gateway and Server in an Indoor High-Precision Positioning System*. In: *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*. Tokyo, Japan: IEEE, 2014-10, pp. 540–541. DOI: 10.1109/GCCE.2014.7031292.

[121] Yang, Z.; Wu, C.; Liu, Y.: *Locating in Fingerprint Space: Wireless Indoor Localization with Little Human Intervention*. In: *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*. Mobicom '12. Istanbul, Turkey: ACM, 2012, pp. 269–280. ISBN: 978-1-4503-1159-5. DOI: 10.1145/2348543.2348578.

[122] Galván-Tejada, C. E.; García-Vázquez, J. P.; Galván-Tejada, J. I.; Delgado-Contreras, J. R.; Brena, R. F.: *Infrastructure-Less Indoor Localization Using the Microphone, Magnetometer and Light Sensor of a Smartphone*. In: *Sensors* 15.8 (2015-08), pp. 20355–20372. ISSN: 1424-8220. DOI: 10.3390/s150820355.

[123] Zhuang, P.; Wang, D.; Shang, Y.: *SMART: Simultaneous Indoor Localization and Map Construction Using Smartphones*. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. Barcelona, Spain: IEEE, 2010-07, pp. 1–8. ISBN: 978-1-4244-6916-1. DOI: 10.1109/IJCNN.2010.5596552.

[124] Jiang, Y.; Pan, X.; Li, K.; Lv, Q.; Dick, R. P.; Hannigan, M.; Shang, L.: *ARIEL: Automatic Wi-Fi Based Room Fingerprinting for Indoor Localization*. In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. UbiComp '12. Pittsburgh, Pennsylvania: ACM, 2012, pp. 441–450. ISBN: 978-1-4503-1224-0. DOI: 10.1145/2370216.2370282.

[125] Martin, E.; Vinyals, O.; Friedland, G.; Bajcsy, R.: *Precise Indoor Localization Using Smart Phones*. In: *Proceedings of the 18th ACM International Conference on Multimedia*. MM '10. Firenze, Italy: ACM, 2010-10, pp. 787–790. ISBN: 978-1-60558-933-6. DOI: 10.1145/1873951.1874078.

[126] Wang, Y.; Zhao, B.; Jiang, Z.: *RSSI-Based Smooth Localization for Indoor Environment*. In: *The Scientific World Journal* 2014 (2014-06), pp. 1–8. DOI: 10.1155/2014/639142.

[127] Chandrasekaran, G.; Ergin, M. A.; Yang, J.; Liu, S.; Chen, Y.; Gruteser, M.; Martin, R. P.: *Empirical Evaluation of the Limits on Localization Using Signal Strength*. In: *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. Rome, Italy: IEEE, 2009-06, pp. 1–9. ISBN: 978-1-4244-2907-3. DOI: 10.1109/SAHCN.2009.5168931.

[128] Park, K.; Shin, H.; Cha, H.: *Smartphone-Based Pedestrian Tracking in Indoor Corridor Environments*. In: *Personal and Ubiquitous Computing* 17.2 (2013-02), pp. 359–370. ISSN: 1617-4917. DOI: 10.1007/s00779-011-0499-5.

[129] Kang, W.; Han, Y.: *SmartPDR: Smartphone-Based Pedestrian Dead Reckoning for Indoor Localization*. In: *IEEE Sensors Journal* 15.5 (2015-05), pp. 2906–2916. ISSN: 1530-437X. DOI: 10.1109/JSEN.2014.2382568.

[130] Subbu, K. P.; Zhang, C.; Luo, J.; Vasilakos, A. V.: *Analysis and Status Quo of Smartphone-based Indoor Localization Systems*. In: *IEEE Wireless Communications* 21.4 (2014-08), pp. 106–112. ISSN: 1536-1284. DOI: 10.1109/MWC.2014.6882302.

[131] *Google: Tango Developer Overview*. URL: https://developers.google.com/tango/developer-overview (visited on 2018-06-30).

[132] Nguyen, K. A.; Luo, Z.: *On Assessing the Positioning Accuracy of Google Tango in Challenging Indoor Environments*. In: *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Sapporo, Japan, 2017-09, pp. 1–8. DOI: 10.1109/IPIN.2017.8115933.

[133] Nilsson, J. O.; Gupta, A. K.; Händel, P.: *Foot-mounted Inertial Navigation Made Easy*. In: *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Busan, South Korea, 2014-10, pp. 24–29. doi: 10.1109/IPIN.2014.7275464.

[134] Endo, Y.; Sato, K.; Yamashita, A.; Matsubayashi, K.: *Indoor Positioning and Obstacle Detection for Visually Impaired Navigation System Based on LSD-SLAM*. In: *2017 International Conference on Biometrics and Kansei Engineering (ICBAKE)*. Kyoto, Japan: IEEE, 2017-09, pp. 158–162. doi: 10.1109/icbake.2017.8090635.

[135] Kamarudin, K.; Mamduh, S. M.; Shakaff, A. Y. M.; Zakaria, A.: *Performance Analysis of the Microsoft Kinect Sensor for 2D Simultaneous Localization and Mapping (SLAM) Techniques*. In: *Sensors* 14.12 (2014-12), pp. 23365–23387. issn: 1424-8220. doi: 10.3390/s141223365.

[136] Martinelli, A.; Tomatis, N.; Siegwart, R.: *Open Challenges in SLAM: An Optimal Solution Based on Shift and Rotation Invariants*. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Vol. 2. New Orleans, LA, USA: IEEE, 2004-04, pp. 1327–1332. isbn: 0-7803-8232-3. doi: 10.1109/ROBOT.2004.1308008.

[137] Schmid, J.; Völker, M.; Gädeke, T.; Weber, P.; Stork, W.; Müller-Glaser, K. D.: *An Approach to Infrastructure-Independent Person Localization with an IEEE 802.15.4 WSN*. In: *2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Zurich, Switzerland: IEEE, 2010-09, pp. 1–9. isbn: 978-1-4244-5862-2. doi: 10.1109/IPIN.2010.5646831.

[138] Pourabdollah, A.; Meng, X.; Jackson, M.: *Towards Low-cost Collaborative Mobile Positioning*. In: *2010 Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*. Kirkkonummi, Finland: IEEE, 2010-10, pp. 1–5. isbn: 978-1-4244-7880-4. doi: 10.1109/UPINLBS.2010.5654023.

[139] Bal, M.; Liu, M.; Shen, W.; Ghenniwa, H.: *Localization in Cooperative Wireless Sensor Networks: A Review*. In: *2009 13th International Conference on Computer Supported Cooperative Work in Design*. Santiago, Chile: IEEE, 2009-04, pp. 438–443. isbn: 978-1-4244-3534-0. doi: 10.1109/CSCWD.2009.4968098.

[140] Mao, G.; Fidan, B.; Mao, G.; Fidan, B.: *Localization Algorithms and Strategies for Wireless Sensor Networks*. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, 2009. isbn: 978-1-60566-396-8.

[141] Karapistoli, E.; Pavlidou, F. N.; Gragopoulos, I.; Tsetsinas, I.: *An Overview of the IEEE 802.15.4a Standard*. In: *IEEE Communications Magazine* 48.1 (2010-01), pp. 47–53. issn: 0163-6804. doi: 10.1109/MCOM.2010.5394030.

[142] Salman, N.; Rasool, I.; Kemp, A. H.: *Overview of the IEEE 802.15.4 Standards Family for Low Rate Wireless Personal Area Networks*. In: *2010 7th International Symposium on Wireless Communication Systems*. York, UK: IEEE, 2010-09, pp. 701–705. isbn: 978-1-4244-6315-2. doi: 10.1109/ISWCS.2010.5624516.

[143] Love, R.: *Linux Kernel Development*. Third Edition. Upper Saddle River, New Jersey, USA: Addison-Wesley Professional, 2010. isbn: 978-0-672-32946-3.

[144] Friesen, J.: *Java XML and JSON*. First Edition. Berkely, CA, USA: Apress, 2016. isbn: 978-1-4842-1915-7. doi: 10.1007/978-1-4842-1916-4.

[145] Leys, C.; Ley, C.; Klein, O.; Bernard, P.; Licata, L.: *Detecting Outliers: Do not use standard deviation around the mean, use absolute deviation around the median.* In: *Journal of Experimental Social Psychology* 49.4 (2013), pp. 764–766. ISSN: 0022-1031. DOI: 10.1016/j.jesp.2013.03.013.

[146] Rousseeuw, P. J.; Croux, C.: *Alternatives to the Median Absolute Deviation.* In: *Journal of the American Statistical Association* 88.424 (1993), pp. 1273–1283. DOI: 10.1080/01621459.1993.10476408.

[147] Efrat, A.; Forrester, D.; Iyer, A.; Kobourov, S. G.; Erten, C.; Kilic, O.: *Force-directed Approaches to Sensor Localization.* In: *ACM Transactions on Sensor Networks (TOSN)* 7.3 (2010-10), 27:1–27:25. ISSN: 1550-4859. DOI: 10.1145/1807048.1807057.

[148] BISWAL, C. P.: *Probability and Statistics.* New Delhi, India: PHI Learning Pvt, 2007. ISBN: 978-81-203-3140-2.

[149] Martinez, W. L.; Martinez, A. R.: *Computational Statistics Handbook with MATLAB.* Third Edition. New York, USA: Taylor & Francis, 2015. ISBN: 978-1-466-59273-5. DOI: 10.1201/b19035.

[150] Allen, M.; Baydere, S.; Gaura, E.; Kucuk, G.: *Evaluation of Localization Algorithms.* In: *Localization Algorithms and Strategies for Wireless Sensor Networks: Monitoring and Surveillance Techniques for Target Tracking.* Ed. by Mao, G.; Fidan, B. IGI Global, 2009, pp. 348–379. ISBN: 978-1-60566-396-8. DOI: 10.4018/978-1-60566-396-8.ch014.

[151] Liu, Y.; Yang, Z.; Wang, X.; Jian, L.: *Location, Localization, and Localizability.* In: *Journal of Computer Science and Technology* 25.2 (2010-03), pp. 274–297. ISSN: 1860-4749. DOI: 10.1007/s11390-010-9324-2.

[152] Yedavalli, K.; Krishnamachari, B.: *Sequence-Based Localization in Wireless Sensor Networks.* In: *IEEE Transactions on Mobile Computing* 7.1 (2008-01), pp. 81–94. ISSN: 1536-1233. DOI: 10.1109/TMC.2008.4387797.

[153] Baggio, A.; Langendoen, K.: *Monte Carlo Localization for Mobile Wireless Sensor Networks.* In: *Ad Hoc Networks* 6.5 (2008-07), pp. 718–733. ISSN: 1570-8705. DOI: 10.1016/j.adhoc.2007.06.004.

[154] Peng, R.; Sichitiu, M. L.: *Robust, Probabilistic, Constraint-Based Localization for Wireless Sensor Networks.* In: *2005 Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005.* Santa Clara, CA, USA: IEEE, 2005-09, pp. 541–550. ISBN: 0-7803-9011-3. DOI: 10.1109/SAHCN.2005.1557106.

[155] Engard, N.: *Practical Open Source Software for Libraries.* Chandos Information Professional Series. Oxford, United Kingdom: Chandos Publishing, 2010. ISBN: 978-1-780-63043-4.

[156] Deek, F. P.; McHugh, J. A.: *Open Source: Technology and Policy.* Cambridge, United Kingdom: Cambridge University Press, 2007. ISBN: 978-0-521-70741-1. DOI: 10.1017/CBO9780511619526.

[157] *Discovery Kit with STM32F407VG MCU.* UM1472. Rev. 5. STMicroelectronics. 2016-02.

[158] *Data Sheet/User Guide PulsON® 440*. 320-0317C. Rev. 1. TIME DOMAIN®. 2016-03.

[159] *STM32F405xx STM32F407xx data sheet*. Rev. 7. STMicroelectronics. 2016-03.

[160] Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P.: *Numerical Recipes: The Art of Scientific Computing*. Third Edition. Cambridge, United Kingdom: Cambridge University Press, 2007. ISBN: 978-0-521-88068-8. DOI: 10.1145/1874391.187410.

[161] Gupta, D. P.: *Design and Analysis of Algorithms*. Second Edition. New Delhi, India: PHI Learning, 2012. ISBN: 978-81-203-4663-5.

[162] Shell, D. L.: *A High-Speed Sorting Procedure*. In: *Communications of the ACM* 2.7 (1959-07), pp. 30–32. ISSN: 0001-0782. DOI: 10.1145/368370.368387.

[163] Sengupta, S.; Korobkin, C. P.: *C++: Object-Oriented Data Structures*. New York, NY, USA: Springer-Verlag New York, 2012. ISBN: 978-1-4612-7618-0. DOI: 10.1007/978-1-4612-2636-9.

[164] Knuth, D. E.: *The Art of Computer Programming, Volume 3: Sorting and Searching*. Second Edition. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1998. ISBN: 0-201-89685-0.

[165] Pradhan, S.; Hwang, S. S.: *Mathematical Analysis of Line Intersection Algorithm for TOA Trilateration Method*. In: *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*. Kitakyushu, Japan: IEEE, 2014-12, pp. 1219–1223. DOI: 10.1109/SCIS-ISIS.2014.7044849.

[166] Silva, B.; Pang, Z.; Åkerberg, J.; Neander, J.; Hancke, G.: *Experimental Study of UWB-based High Precision Localization for Industrial Applications*. In: *2014 IEEE International Conference on Ultra-WideBand (ICUWB)*. Paris, France: IEEE, 2014-09, pp. 280–285. DOI: 10.1109/ICUWB.2014.6958993.

[167] Beutel, J.: *Geolocation in a PicoRadio Environment*. MA thesis. ETH Zurich, Department of Electrical Engineering, 1999.

[168] Norrdine, A.: *An Algebraic Solution to the Multilateration Problem*. In: *The Third International Conference on Indoor Positioning and Indoor Navigation (IPIN2012)*. Sydney, Australia, 2012-04, pp. 1–5.

[169] Nocedal, J.; Wright, S.: *Numerical Optimization*. New York, NY, USA: Springer New York, 2006. ISBN: 978-0-387-40065-5. DOI: 10.1007/978-0-387-40065-5.

[170] Daubechies, I.: *Ten Lectures on Wavelets*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992. ISBN: 978-0-89871-274-2. DOI: 10.1137/1.9781611970104.

[171] Noble, B.; Daniel, J.: *Applied Linear Algebra*. Third Edition. Prentice-Hall, 1988. ISBN: 978-0-13-041260-7.

[172] Leach, S.: *Singular Value Decomposition – A Primer*. In: *Unpublished Manuscript, Department of Computer Science, Brown University, Providence, RI, USA* (1995).

[173] Neapolitan, R.: *Foundations of Algorithms*. Fifth Edition. Sudbury, Massachuetts, USA: Jones & Bartlett Learning, 2015. ISBN: 978-1-284-04919-0.

[174] Martinez, D. R.; Bond, R. A.; Vai, M. M.: *High Performance Embedded Computing Handbook: A Systems Perspective*. Boca Raton, Florida, USA: CRC Press, 2008. ISBN: 978-0-8493-7197-4. DOI: 10.1201/9781315221908.

[175] Smoktunowicz, A.; Wróbel, I.: *Numerical Aspects of Computing the Moore-Penrose Inverse of Full Column Rank Matrices*. In: *BIT Numerical Mathematics* 52.2 (2012-06), pp. 503–524. ISSN: 1572-9125. DOI: 10.1007/s10543-011-0362-0.

[176] Blankenbach, J.; Norrdine, A.: *Position Estimation Using Artificial Generated Magnetic Fields*. In: *2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Zurich, Switzerland: IEEE, 2010-09, pp. 1–5. DOI: 10.1109/IPIN.2010.5646739.

[177] Chung, J.; Donahoe, M.; Schmandt, C.; Kim, I.-J.; Razavai, P.; Wiseman, M.: *Indoor Location Sensing Using Geo-Magnetism*. In: *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*. MobiSys '11. Bethesda, Maryland, USA: ACM, 2011, pp. 141–154. ISBN: 978-1-4503-0643-0. DOI: 10.1145/1999995.2000010.

[178] Carrella, S.; Kuncup, I.; Lutz, K.; König, A.: *3D-Localization of Low-Power Wireless Sensor Nodes Based on AMR-Sensors in Industrial and AmI Applications*. In: *Konferenz: Sensoren und Messsysteme 2010*. ITG/GMA-Fachtagung. Nürnberg, Germany: VDE Verlag GmbH Berlin Offenbach, 2013.

[179] Blankenbach, J.; Norrdine, A.; Hellmers, H.: *A Robust and Precise 3D Indoor Positioning System for Harsh Environments*. In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Sydney, NSW, Australia: IEEE, 2012-11, pp. 1–8. ISBN: 978-1-4673-1954-6. DOI: 10.1109/IPIN.2012.6418863.

[180] *Smart Digital Magnetometer HMR2300*. Rev. J. Honeywell International Inc. 2012-01.

[181] *LPC2387 Product Data Sheet*. Rev. 5.1. NXP Semiconductors. 2013-10.

[182] Van Valkenburg, M. E.: *Reference Data for Engineers: Radio, Electronics, Computers and Communications*. Ninth Edition. Newnes, 2002. ISBN: 978-0-7506-7291-7. DOI: 10.1016/B978-0-7506-7291-7.50054-6.

[183] Vig, John R.: *Quartz Crystal Resonators and Oscillators for Frequency Control and Timing Applications*. In: *Learning Resources*. IEEE Ultrasonics, Ferroelectrics, and Frequency Control, 2014. URL: https://ieee-uffc.org/frequency-control/educational-resources/introduction-to-quartz-frequency-standards-by-john-r-vig (visited on 2019-07-15).

[184] Zhou, H.; Nicholls, C.; Kunz, T.; Schwartz, H.: *Frequency Accuracy & Stability Dependencies of Crystal Oscillators*. Tech. rep. SCE-08-12. Ottawa, Ont., Canada: Department of Systems and Computer Engineering, Carleton University, 2008-11, pp. 1–15.

[185] Kumar, A.; Madaan, P.: *How to Generate a Precise Clock Source*. EDN Network. 2012-06. URL: https://www.edn.com/design/sensors/4406691/3/Oscillators--How-to-generate-a-precise-clock-source (visited on 2019-07-15).

[186] Westra, J. R.; Verhoeven, C. J.; van Roermund, A.: *Oscillators and Oscillator Systems: Classification, Analysis and Synthesis*. Boston, MA, USA: Springer US, 1999. ISBN: 978-1-4419-5110-6. DOI: 10.1007/978-1-4757-6117-7.

[187] Litayem, N.; Saoud, S. B.: *Impact of the Linux Real-time Enhancements on the System Performances for Multi-core Intel Architectures*. In: *International Journal of Computer Applications* 17.3 (2011-03). Full text available, pp. 17–23. doi: 10.5120/2202-2796.

[188] Buttazzo, G.; Lipari, G.: *Ptask: an Educational C Library for Programming Real-Time Systems on Linux*. In: *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*. Cagliari, Italy: IEEE, 2013-09, pp. 1–8. doi: 10.1109/ETFA.2013.6648001.

[189] STMicroelectronics: *Clock/Calendar Implementation on the STM32F10xxx Microcontroller RTC*. STMicroelectronics Application Note (AN2821). Rev 2. 2009-04.

[190] Torres, S.: *Using Processor Expert to Develop a Software Real-Time Clock*. Freescale Semiconductor Application Note (AN2949). Rev 1. 2006.

[191] Jiménez, M.; Palomera, R.; Couvertier, I.: *Introduction to Embedded Systems: Using Microcontrollers and the MSP430*. New York, NY, USA: Springer New York, 2014. isbn: 978-1-4614-3143-5. doi: 10.1007/978-1-4614-3143-5.

[192] Steven Bible: *Crystal Oscillator Basics and Crystal Selection for rfPIC and PICmicro Devices*. Microchip Technology Inc. Application Note (AN826). Rev 1. 2002.

[193] Sergio Garcia de Alba Garcin: *Crystal Oscillator Troubleshooting Guide*. Freescale Semiconductor Application Note (AN3208). Rev 0. 2006-01.

[194] STMicroelectronics: *Oscillator Design Guide for ST Microcontrollers*. Application Note (AN2867). Rev 5. 2011-03.

[195] *Extremely Accurate SPI Bus RTC with Integrated Crystal and SRAM (DS3234)*. 19-5339. Rev. 4. Maxim Integrated. San Jose, CA, USA, 2015-03.

[196] Lenk, A.; Ballas, R. G.; Werthschützky, R.; Pfeifer, G.: *Electromechanical Systems in Microtechnology and Mechatronics: Electrical, Mechanical and Acoustic Networks, their Interactions and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. isbn: 978-3-642-10805-1. doi: 10.1007/978-3-642-10806-8.

[197] Björck, A.: *Numerical Methods for Least Squares Problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1996. isbn: 978-0-898-71360-2. doi: 10.1137/1.9781611971484.

[198] Blankenbach, J.; Norrdine, A. In: Karimi, H. A. *Indoor Wayfinding and Navigation*. First Edition. CRC Press, Inc., 2015. Chap. Magnetic Indoor Local Positioning System, pp. 53–80. isbn: 978-1-4822-3084-0. doi: 10.1201/b18220.

[199] Dahmen, W.; Reusken, A.: *Numerik für Ingenieure und Naturwissenschaftler*. Berlin, Heidelberg: Springer-Verlag, 2008. isbn: 978-3-540-76492-2. doi: 10.1007/978-3-540-76493-9.

[200] *Real-Time Clock Calculator*. Maxim Integrated, San Jose, CA, USA. url: https://www.maximintegrated.com/en/design/tools/calculators/product-design/rtc.cfm (visited on 2019-07-15).

[201] Sheinker, A.; Ginzburg, B.; Salomonski, N.; Frumkis, L.; Kaplan, B. Z.: *Localization in 2D Using Beacons of Low Frequency Magnetic Field*. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 6.2 (2013-04), pp. 1020–1030. issn: 1939-1404. doi: 10.1109/JSTARS.2012.2213240.

[202] Nixon, M. A.; McCallum, B. C.; Fright, W. R.; Price, N. B.: *The Effects of Metals and Interfering Fields on Electromagnetic Trackers*. In: *Presence: Virtual and Augmented Reality* 7.2 (1998-04), pp. 204–218. ISSN: 1054-7460. DOI: 10.1162/105474698565587.

[203] Rose, J. A.; Tong, J. R.; Allain, D. J.; Mitchell, C. N.: *The Use of Ionospheric Tomography and Elevation Masks to Reduce the Overall Error in Single-Frequency GPS Timing Applications*. In: *Advances in Space Research* 47.2 (2011-06), pp. 276–288. ISSN: 0273-1177. DOI: 10.1016/j.asr.2010.05.030.

[204] Madsen, K.; Nielsen, H. B.; Tingleff, O.: *Methods for Non-Linear Least Squares Problems (Second Edition)*. Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, Denmark, 2004.

[205] Datta, B. N.: *Numerical Linear Algebra and Applications, Second Edition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2010. ISBN: 978-0-89871-685-6.

[206] Demmel, J. W.: *Applied Numerical Linear Algebra*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997. ISBN: 0-89871-389-7. DOI: 10.1137/1.9781611971446.

[207] Golub, G. H.; Van Loan, C. F.: *Matrix Computations (4th Edition)*. Baltimore, MD, USA: Johns Hopkins University Press, 2013. ISBN: 978-1-4214-0794-4.

[208] Trefethen, L. N.; Bau, D.: *Numerical Linear Algebra*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics (SIAM), 1997. ISBN: 978-0-89871-361-9.

[209] Strang, G.: *Linear Algebra and Its Applications*. Fourth Edition. Belmont, CA, US: Thomson, Brooks/Cole, 2006. ISBN: 978-0-03-010567-8.

[210] Hogben, L.: *Handbook of Linear Algebra*. Second Edition. Boca Raton, Florida, USA: CRC Press/Taylor & Francis Group, 2016. ISBN: 978-1-138-19989-7. DOI: 10.1201/b16113.

[211] Mittrapiyanuruk, P.: *A Memo on How to Use the Levenberg-Marquardt Algorithm for Refining Camera Calibration Parameters*. Robot Vision Laboratory, Purdue University. West Lafayette, IN, USA, 2006.

[212] Schröder, D.: *Intelligente Verfahren: Identifikation und Regelung nichtlinearer Systeme*. Springer-Verlag Berlin Heidelberg, 2010. ISBN: 978-3-642-11397-0. DOI: 10.1007/978-3-642-11398-7.

[213] Schröder, D.: *Intelligent Observer and Control Design for Nonlinear Systems*. Springer Berlin Heidelberg, 2000. ISBN: 978-3-642-08346-4. DOI: 10.1007/978-3-662-04117-8.

[214] Denis, B.; Keignart, J.; Daniele, N.: *Impact of NLoS Propagation upon Ranging Precision in UWB Systems*. In: *IEEE Conference on Ultra Wideband Systems and Technologies*. Reston, VA, USA: IEEE, 2003-11, pp. 379–383. ISBN: 0-7803-8187-4. DOI: 10.1109/UWBST.2003.1267868.

[215] Prieto, J. C.; Croux, C.; Jiménez, A. R.: *RoPEUS: A New Robust Algorithm for Static Positioning in Ultrasonic Systems*. In: *Sensors* 9.6 (2009-06), pp. 4211–4229. ISSN: 1424-8220. DOI: 10.3390/s90604211.

[216] Rousseeuw, P. J.; Leroy, A. M.: *Robust Regression and Outlier Detection*. New York, NY, USA: John Wiley & Sons, Inc., 1987. ISBN: 978-0-471-85233-9. DOI: 10.1002/0471725382.

[217] Sol Kim, A.; Hwang, J.; Park, J.: *Enhanced Indoor Positioning Algorithm Using WLAN RSSI Measurements Considering the Relative Position Information of AP Configuration*. In: *Journal of Institute of Control, Robotics and Systems* 19.2 (2013-02), pp. 146–151.

[218] Zou, C.; Kim, A.; Hwang, J.; Park, J.: *Enhanced Positioning Method Using WLAN RSSI Measurements Considering Dilution of Precision of AP Configuration*. In: *Seventh International Conference on Systems and Networks Communications 2012 (ICSNC 2012)*. Lisbon, Portugal: International Academy, Research, and Industry Association (IARIA), 2012, pp. 186–190. ISBN: 978-1-62276-585-0.

[219] Alavi, B.; Pahlavan, K.: *Modeling of the TOA-based Distance Measurement Error Using UWB Indoor Radio Measurements*. In: *IEEE Communications Letters* 10.4 (2006-04), pp. 275–277. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2006.1613745.

[220] *Raspberry Pi 3 Model B*. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b (visited on 2019-07-15).

[221] Guerchali, N.: *Untersuchung des Levenberg-Marquardt-Algorithmus zur Indoor-Lokalisierung für den STMF407-Mikrocontroller*. 2017.