Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Hyperbolic positioning in UWB networks with non-transmitting tag

Master's Thesis

Bc. Josef Krška

Prague, May 2021

Study program:	Cybernetics and Robotics
Branch of study:	Cybernetics and Robotics
Supervisor:	Ing. Václav Navrátil, Ph.D.

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Krška Josef**	Personal ID number:	**457187**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Hyperbolic positioning in UWB networks with non-transmitting tag

Master’s thesis title in Czech:

Hyperbolické určování polohy v UWB sítích s nevysílajícím tagem

Guidelines:

The subject of the diploma thesis is to design, describe and implement a tag localization by means of Time difference of arrival (TDoA) method within the network of synchronized UWB anchors. Assume that anchors transmit signals and the tag acts only as a receiver. Focus on the effect of local clock drift upon the measurements. Also, explore the various options of anchor localization and possibilities of connection to the coordinate systems used by GNSS.

Bibliography / sources:

[1] Z. Sahinoglu, S. Gezici, and I. Gvenc, Ultra-wideband Positioning Systems: Theoretical Limits, Ranging Algorithms, and Protocols. New York, NY, USA: Cambridge University Press, 2011, isbn: 978-0-521-18783-1.
[2] Z. Koppanyi, C. K. Toth and D. G. Brzezinska, "Scalable ad-hoc UWB network adjustment," 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS), Monterey, CA, 2018, pp. 1502-1508, doi: 10.1109/PLANS.2018.8373544.
[3] P. Misra and P. Enge, Global Positioning System: Signals, Measurements, and Performance, G.-J. Press, Ed. Lincoln, Mass.: Ganga-Jamuna Press, 2001, isbn: 0-9709544-0-9.

Name and workplace of master’s thesis supervisor:

Ing. Václav Navrátil, Ph.D., Department of Radioelectronics, FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: ___________ Deadline for master's thesis submission: **21.05.2021**

Assignment valid until:
by the end of summer semester 2021/2022

Ing. Václav Navrátil, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt — Student’s signature

Abstract

In common TDoA-UWB networks the user device broadcasts a message, which is received by the surrounding infrastructure. The gathered data is then used to determine the position of the user. Therefore, the estimated position is immediately available to the infrastructure only. For the purposes of navigation, it is necessary to deliver the estimates to the users as soon as possible.

The thesis aims to develop a TDoA-UWB system, where users are able to estimate their own positions. To achieve it, the infrastructure broadcasts messages and the user devices act as passive listeners. Such approach faces a challenge related to the clock drift of the user devices. A solution, which exploits the Extended Kalman filter and achieves low estimation errors, is developed within the thesis.

The use of distributed computing for the self-estimation of the network infrastructure node positions is investigated as well. The presented algorithms allow the network nodes to estimate their positions based on the mutual distance measurements and several initial node positions.

Keywords

indoor positioning, navigation, position estimation, ultra wideband, time difference of arrival, extended Kalman filter, passive tag, network adjustment, consensus subgradient

Abstrakt

V běžných TDoA-UWB sítích uživatelské zařízení vysílá zprávu, která je přijata okolní infrastrukturou. Poloha zařízení je určena na základě shromážděných dat z této zprávy. Ihned po dokončení odhadu polohy je výsledek znám pouze infrastruktuře. Avšak pro účely navigace je nutné doručit odhad polohy co nejdříve i uživateli.

Tato práce se soustředí na vývoj TDoA-UWB systému, který umožňuje uživatelům odhadnout svou vlastní polohu. Toho je dosaženo použitím infrastruktury k vysílání zpráv a změnou uživatelských zařízení na pasivní přijímače. Tento přístup však musí řešit problém spojený s driftem interních hodin uživatelských zařízení. Navrhované řešení využívá rozšířeného Kálmánova filtru a dosahuje nízkých chyb odhadu polohy.

Část práce se také zabývá využitím distribuovaných výpočtů pro samostatný odhad polohy uzlů síťové infrastruktury. Diskutované algoritmy umožňují uzlům odhadnout vlastní polohy pomocí naměřených vzájemných vzdáleností a několika počátečních poloh.

Klíčová slova

indoor lokalizace, navigace, odhad polohy, ultra wideband, time difference of arrival, rozšířený Kálmánův filtr, pasivní tag, optimalizace sítě, consensus subgradient

Acknowledgments

I would like to thank prof. Ing. František Vejražka, CSc. and Ing. Václav Navrátil Ph.D. for their guidance, patience, advice, and insight. I would also like to thank Ing. Jan Cabicar for rectifying my English creativity.

My special thanks goes to my family and Hela for their infinite support, patience and that they had the strength to get along with me while I was working on my thesis. Without them it would be much more difficult to finish it.

Thank you all.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaboratingan academic final thesis

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

20 May 2021, Prague

. . . .
Student's signature

Contents

1 Introduction

When it comes to the localization of an entity, being that a car, pedestrian, goods or other movable asset, the common first choice is any system from the Global Navigation Satellite System (GNSS) group or their combination. The systems like the Global Positioning System (GPS), GLONASS, BeiDou or the European Galileo are considered to be GNSS. The GNSS offers relatively high accuracy of localization of the user devices, ranging from tens of centimeters to meters. However such precision is achievable in the outdoor environments only, where the user device has an unobstructed view of the sky. In the urban or indoor environments the localization precision quickly deteriorates and therefore a localization system that does not utilize the satellites has to be used.

A number of alternative localization systems, suitable for GNSS-denied environments, already exists, achieving varying accuracy. For example positioning using Wi-Fi networks and power measurements (RSSI) has typical accuracy of several meters [1].

Among the most promising alternative positioning systems are the ones using the Ultra Wide Band (UWB) signals. The UWB signals are governed by the IEEE 802.15.4 standard [2] and utilize spread spectrum with very wide bandwidth (e.g. 500 MHz) in order to achieve a precise time measuring with resolution in tens of picoseconds [2].

In our previous works [3, 4] we have developed a UWB localization network, which uses the *Time Difference of Arrival* (TDoA) localization principle for the data acquisition and the actual position estimation.

It is common for the UWB-TDoA localization networks that the user device, so called *tag*, periodically broadcasts a *blink* message. Such message is received by the surrounding *anchors* and the tag's position is determined based on the data related to that message. With this *Tag to Anchor* (T2A) TDoA variant, the resulting position is available only at the computing center and not to the user. The system developed in [3, 4] uses the same T2A messaging scheme.

While the T2A-TDoA is great for the tracking of equipment or vehicles in the pursuit of optimal transportation, it is not suitable for cases when there are many users and each user needs his position immediately.

In this work we aim to develop the *Anchor to Tag* (A2T) TDoA localization system, which makes the position estimates available to the users immediately after their computation and also allows an infinite number of users to be positioned simultaneously. This is achieved by reversing the direction of the messages, making the anchors periodically send localization messages and the tags passive listeners.

As this work builds on [3, 4] we recapitulate those works in Chapters 1 and 2. In Chapter 1 we introduce the principles of the UWB communication and localization. Chapter 2 then reminds the concepts and implementation of the T2A-TDoA together with the addition of the Chained synchronization algorithm. Chapter 3 gives a proper introduction to the problem of A2T-TDoA, what challenges are faced and finally the implementation of the A2T-TDoA with the experimental results. The last Chapter 4 stands aside from the topic of the user localization as it focuses on the problem of the UWB network installation and determination of the anchor positions. The chapter provides a semi-automatic algorithm for anchor self-positioning in larger networks.

This work also includes two appendices. Appendix A discusses the computation of the *Dilution of Precision* (DOP) parameter and Appendix B presents the results of the previously developed T2A-TDoA UWB network.

1.1 Ultra Wide Band localization system

In this section we provide a description of what an Ultra Wide Band (UWB) signal is and from which devices the UWB localization network consists of. First we define a UWB signal and then introduce a specific device that uses the UWB signals for communication and measurements.

1.1.1 Ultra Wide Band signals

An *Ultra Wide Band* signal can be described in a broad sense as a signal having the absolute bandwidth larger than 500 MHz or the relative bandwidth of at least 20 % [5] (relative bandwidth is a ratio of the absolute bandwidth and the center frequency). In the narrower sense, we see ultra wide band signals as a group of signals that is defined by the IEEE 802.15.4 standard [2] where definition of both the physical layer and device communication (media access) is provided. Our main interest is in the variant of UWB physical layer called *High Rate Pulse Repetition Frequency* UWB physical layer (HRP UWB PHY) [2], so from this point forward we simply use the short UWB term instead of HRP UWB PHY.

The standard [2] specifies 16 UWB channels (numbered from 0 to 15), grouped into 3 bands. One channel in the sub-gigahertz band (from 250 MHz to 750 MHz), 4 channels in the lower band (3.244 GHz to 4.742 GHz) and 11 channels in the high band (5.944 GHz to 10.234 GHz). Most of the channels have 499.2 MHz bandwidth, nonetheless, channels 4, 7, 11 and 15 have bandwidth wider than 1 GHz.

The standard was created with an intended use in wireless personal area networks (WPAN), comprising of low-energy devices with low communication rates. Data within a network is transmitted using a series of very short pulses (typically shorter than a nanosecond [5]), meaning that there is not a carrier wave that carries the information as opposed to a typical narrowband technologies such as GSM or Wi-Fi.

The usage of the impulses for the information transmission is the reason for the distinctive property of the UWB, the very large bandwidth. The utilization of the impulses has several advantages. One is the low power consumption as the UWB devices do not transmit energy continuously during the transmission of a message, but rather in discrete impulses, which is less power consuming. The second and probably main advantage of the UWB is again its bandwidth. Not only that it enables bit rates up to 27.24 Mbps, according to [2], but it also makes the UWB highly suitable for positioning.

Let us consider a simple measurement of *time of arrival* (ToA) of a message at the device's antenna. When using UWB signals we can achieve a measurement with a *very low variance*, especially in comparison with narrowband signals (under the assumption of sufficiently high signal to noise ratio). The possibility of accurate measurement comes directly from the Cramér-Rao lower bound (CRLB) for ToA measurements via UWB signals [5, 6]. This lower bound determines the lowest theoretically achievable variance by an unbiased estimator of a deterministic parameter.

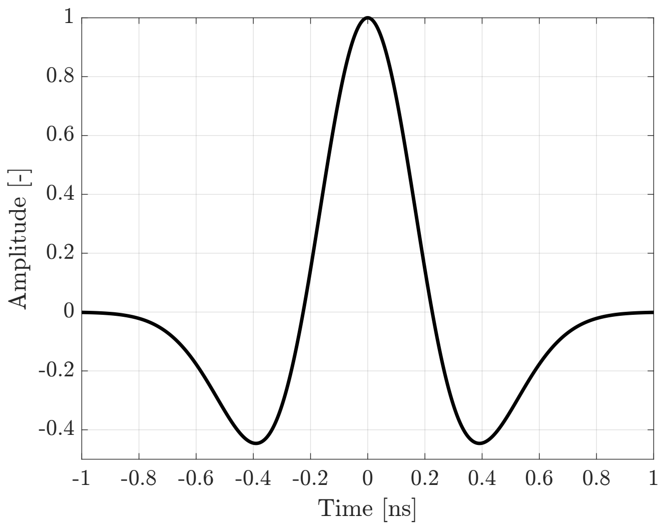


Figure 1.1: UWB pulse with pulse width $T_p = 0.8\,\mathrm{ns}$

The analysis of the ToA measurement's CRLB together with its derivation is done in [5, 6] and here we will use the analysis result only

$$\sqrt{\sigma_t^2} \geq \frac{1}{2\pi\sqrt{\mathrm{SNR}}\,\beta}, \tag{1.1}$$

where t is the measured ToA, σ_t^2 is its variance, SNR is signal to noise ratio and β is the effective bandwidth of the measuring signal (for its definition refer to [5]). The effective bandwidth is difficult to estimate precisely as it varies for each signal and heavily depends on the traveled environment. Nonetheless, it is often approximated by the 3 dB [6] signal bandwidth (e.g. 500 MHz).

The impulses used in the communication often involve shapes of derivatives of the Gaussian pulse, wavelet pulses and modified Hermite polynomials [5]. For its usage of impulses, the UWB is also known as the Impulse Radio UWB or IR-UWB. In Figure 1.1 we can see an example of a simple UWB pulse with pulse width of 0.8 ns.

The actual pulses used for communication are much more complex than the one in Figure 1.1. Nevertheless, it is useful for getting the notion of their appearance. The pulse width of a UWB signal can be viewed as a parameter which affects the resulting bandwidth and thus the CRLB. We can use the value of the pulse width to estimate the effective bandwidth of a raised-cosine pulse [5, 6] and put it in the Equation (1.1) and then plot how the CRLB changes with changing SNR.

In Figure 1.2 we can see the plotted CRLB of the ToA measurement variance for three pulse widths. The right axis in Figure 1.2 shows the standard deviation of the time measurement and the left axis the standard deviation of the range measurement (time standard deviation multiplied by the propagation speed c). We can see that even for SNR equal to 10 dB the measurement standard deviation can already be lower than 10 cm, which makes the measurements usable for the positioning even for low SNR. In practice we expect the SNR to be higher than 20 dB, so the measurement standard deviation is reduced even further.

For communication the UWB uses the BPM-BPSK (Burst Position modulation, Binary Phase Shift Keying) modulation, in which the smallest unit of transmitted information is called a symbol [2].

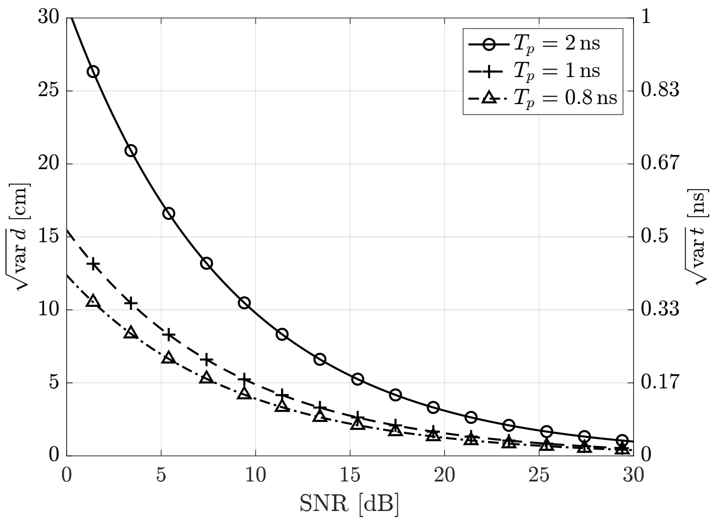


Figure 1.2: Cramér-Rao Lower Bound for selected pulse widths [6]

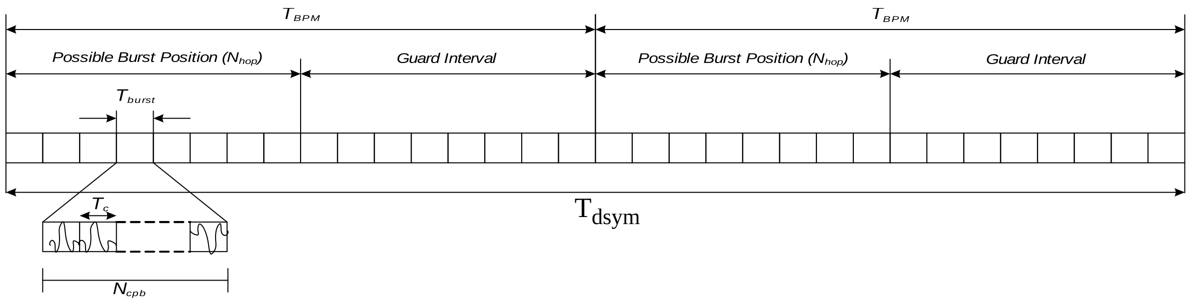


Figure 1.3: Structure of an UWB symbol [2]

Figure 1.3 shows the structure of a UWB symbol. Each symbol carries 2 bits of information and each symbol is divided into 4 time intervals, 2 intervals called *Possible Burst Position* and 2 *Guard* intervals. Only one UWB impulse (or impulse burst) can occur within a single symbol in either of the two possible burst position intervals. This position encodes one bit of the information while the second bit is determined by the phase of the burst.

During guard intervals no burst is allowed to be transmitted, which increases resistance to the multipath effects. The reason is that if any burst is detected during a guard interval, it cannot be a valid burst of the signal of interest and it must be either a reflection or originate from a different transmitter. Such invalid burst can be ignored and does not interfere with the ongoing communication.

Although having wide bandwidth brings desirable advantages it also brings certain disadvantages. The transmitted power of a UWB device is a subject of the regulations [2]. The regulations also impose a maximal radiated power spectral density (PSD) or maximal continuous transmission. For example in the band from 3.4 GHz to 4.8 GHz (channels 1, 2, 3 and 4) the maximum mean PSD is $-41.3\,\mathrm{dBm/MHz}$. However if the UWB device is used by emergency services, this limit is lifted up to $-21.3\,\mathrm{dBm/MHz}$ [7, 8].

These limits reduce the maximal achievable range of the UWB signals. Also, the propagation through the walls and obstacles is poor due to the use of high frequencies.

On the other hand, the UWB signals are suitable for the indoor environments as the range there is limited as well. This makes the UWB a great complement to the GNSS, whose indoor performance is generally considered poor.

1.1.2 UWB network

In the previous section, we have introduced the UWB signals and their key properties. As we have seen, a device implementing the IEEE 802.15.4 standard is able to precisely measure the time of arrival of a UWB signal at the device's antenna.

For the purposes of localization using UWB signals we have designed an UWB localization network [3, 4], where the UWB communication is provided by the DW1000 chip developed by the Decawave Ltd. company. This chip served as a base for the design of the network nodes that form the network infrastructure and the user devices within the network. We will discuss these devices later in this section, nonetheless, we will describe the DW1000 chip first.

The DW1000 transceiver chip implements the physical layer of the UWB communication in compliance with the IEEE 802.15.4 standard [2]. The chip is able to measure the time of reception (arrival) and transmission of the IR-UWB messages. Both of these measurements are denoted simply as a message time-stamping (regardless of the message direction).

The time-stamping process relies on an on-chip PLL-based clock with the nominal frequency of 63.8976 GHz[1], as defined by the IEEE 802.15.4, providing us with approximately 15.65 ps or 4.7 mm resolution. The frequency is an integer multiple of the 38.4 MHz crystal oscillator or external reference. However, the time-stamping process is not perfect and every measurement has the standard deviation approximately equal[2] to 150 ps (or 4.5 cm) [9]. The time-stamping frequency is used to increment the internal counter, which can be viewed as a free running clock associated with the specific device. According to the standard [2] the counter is only 32-bit wide, in which case the overflow event happens approximately every 67 ms. While this interval is sufficiently long for positioning and time measurements, some manufacturers chose to use a wider counter. In case of the DW1000 this counter features width of 40 bits, which extends the overflow period to 17.2 s.

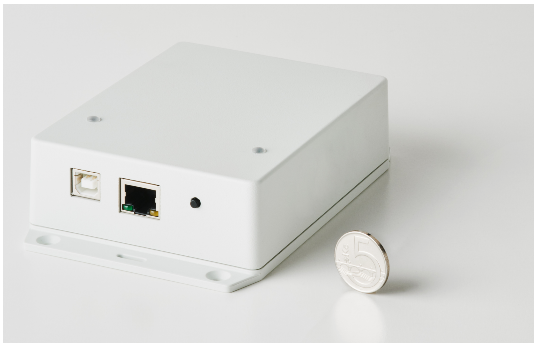
The infrastructure of a UWB network consists of the **anchors**, nodes with fixed and *a priori* known position. The anchor device can be seen in Figure 1.4a. Apart from the DW1000 UWB chip the anchor embeds two additional communication interfaces, the USB and the Ethernet (with PoE). The PoE capability reduces the number of needed cable connections by utilizing the unused data wires in the Ethernet cable for power delivery which makes the installation and mounting more flexible. Anchor also has an embedded battery which can be charged using any of the two available power sources (USB or PoE).

Tags are small, handheld devices meant to be the user devices and can be seen in Figure 1.4b. Position of such a device is not *a priori* known and is rather estimated using the localization network. Tag is usually powered by a built-in battery so its power consumption has to be as low as possible in order to maximize the operational time between recharges. Unlike the anchor, the tag possesses only USB interface, which is used for configuration and battery charging. The tag also uses the DW1000 chip for UWB communication.

Inside of the tags are also MEMS inertial sensors, whose outputs can be used for estimating the device's orientation. Especially orientation or movement detection can be useful for power

[1]For the sake of briefness we use the approximate value 64 GHz.

[2]The measurement standard deviation of 150 ps is valid for signals with high SNR. With lower SNR the deviation increases.


(a) Anchor

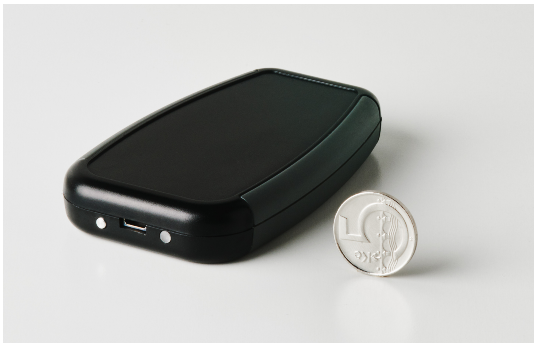

(b) Tag

Figure 1.4: UWB localization devices

management, as the tag can adjust its position fix frequency depending on the nature and velocity of its movement.

Both the tag and the anchor devices participate in data exchange within the network in order to estimate the position of the tag. In previous works [3, 4] the used localization principle was the TDoA in the variant, where the tags periodically send *blink* messages that are received by the anchors. Based on the direction of the message transmission, we have named this implementation as the *Tag to Anchor* TDoA (T2A-TDoA). Description and the implementation of T2A-TDoA has been the main aim of our previous works [3] and we will recapitulate this topic in Chapter 2.

In this work however we mainly focus on the other option called *Anchor to Tag* TDoA, where the anchors transmit messages and tags receive them. This TDoA variant will be described in Chapter 3.

It should be also noted that the hardware for both the anchor and tag has been developed and manufactured in the cooperation of the company RCD Radiokomunikace[3] and the Department of Radio Engineering from the Faculty of Electrical Engineering on Czech Technical University in Prague.

The UWB localization network also includes another node type. The **Computation node** serves as a central data collection point of any data shared in the network. This node can be any computer connected to the UWB network, preferably by an Ethernet connection. It is mostly used in the T2A TDoA, where the anchors send data about the received blink messages to this central node. The task of the node is to estimate the positions of the tags from the received data and store them, eventually to provide them for visualization.

1.2 Suitable positioning principles

As we have emphasized in the previous sections, the UWB signals according to the Cramér-Rao Lower Bound offer a great precision when used for time measuring or message time-stamping [6]. These measurements can be used for the estimation of the user position.

Given the accurate time measurements the suitable localization principles are those that involve ranges between the tags and anchors during the estimation. In the UWB localization networks the two most used principles are the *Time of Flight* (ToF), implemented using the *Two Way Ranging* (TWR) protocol, and the *Time Difference of Arrival* (TDoA).

[3]Based in Staré Hradiště, Czech Republic.

Both TWR and TDoA principles incorporate distances, or more precisely the times of flight, between the tag and anchors to estimate the tag's position. Because of the free-running nature of the device clocks (both the anchors and tags), there exists a bias between the time bases of any two anchors. Moreover this bias is not constant and changes in time due to the instability of the reference oscillator frequency.[4] In our experiments we have also observed that the bias change, or bias drift, is not constant either.

The bias evolution can be captured by the following non-linear model

$$b_i(t) = b_{i0} + \int_0^t \left(\dot{b}_i(\tau) + \int_0^\tau \left[\ddot{b}_i(s)\,\mathrm{d}s \right] \mathrm{d}\tau \right) , \tag{1.2}$$

where $b_i(t)$ is the current bias between the time base of the anchor i and reference time base, b_{i0} is the initial value of bias, $\dot{b}_i$ is their mutual bias drift and $\ddot{b}_i$ is their mutual bias drift change rate. Note, that we are only interested in the *relative* bias between two chosen time bases rather than an absolute time bias (w.r.t. UTC time, for instance).

In practice it is too difficult to determine the nonlinear model behind the bias dynamics with the precision needed. It is much more feasible to get bias measurements in regular intervals and, if needed, estimate the bias value between the measurements by a simplified and discretized model. If we assume that the bias changes slowly over a short period of time, then we can estimate the bias value between measurements with a discrete linear model

$$b_i[k+1] = b_i[k] + \dot{b}_i[k]T_e + \frac{1}{2}\ddot{b}_i[k]T_e^2 , \tag{1.3}$$

where the T_e is the time delay between the last measurement at time k and the estimation.

Measurements conclude that for reasonably long intervals between consecutive timing events (T_e) this linear model does not induce big errors that would harm the position estimation precision [10, 11].

The presence of bias in the measurements is an unwanted error source, which in the context of distance measurements corrupts the measurements so that the resulting distances are either too large or negative. For example if we have two oscillators, both having the bias drift in the range ± 20 ppm, then in the worst case their mutual bias drift is ± 40 ppm. With each passed millisecond the mutual bias will grow approximately by 40 ns which results in the ranging error of at least 12 m if left uncompensated.

As both localization principles are challenged by the bias dynamics, both of them must have the means to cope with it. Table 1.1 gives a quick comparison of the discussed localization principles, the TWR, T2A-TDoA and A2T-TDoA. The following sections will discuss the properties of those principles.

1.2.1 Two Way Ranging

The Two Way Ranging protocol is an implementation of the *time of flight* (ToF) principle. Before we describe the TWR we will talk about a general ToF principle.

The time of flight principle estimates the user position using distances measured between the tag and anchors. A single distance measurement restricts the set of possible tag positions to a circle around the particular anchor. In the three dimensions that set is a sphere. In total, three different measurements from three different anchors and specification of the half-space

[4]Typically, the oscillator frequency is mostly (but by far not exclusively) affected by device internal temperature. Consequently, the drift change is commonly observed after device start-up.

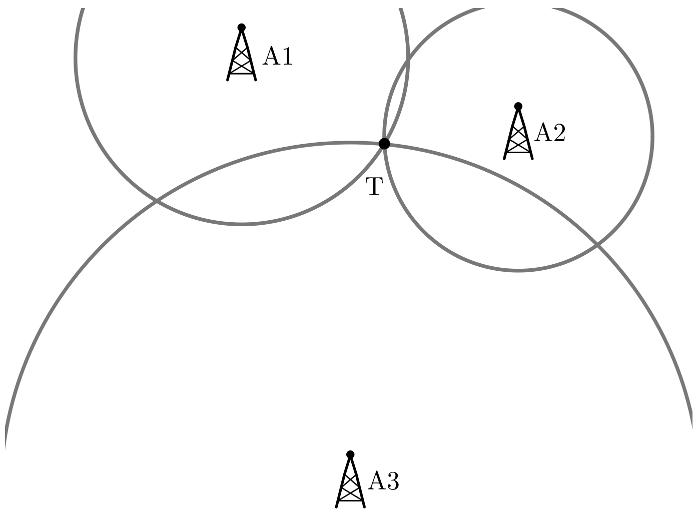


Figure 1.5: Time of Flight positioning principle

are needed to find an unambiguous three-dimensional solution for the tag position, which lies in the intersection of the spheres. An example of ToF positioning with three anchors in two dimensions is shown in Figure 1.5.

If the speed of signal propagation in the area is (almost) constant and known, then the distance between the tag and anchor can be calculated from the transmission and reception times of messages sent between the devices.

We may conclude that for each transmission t_{Tx} and reception time t_{Rx} of a message sent between tag and anchor (regardless of the direction) the following holds

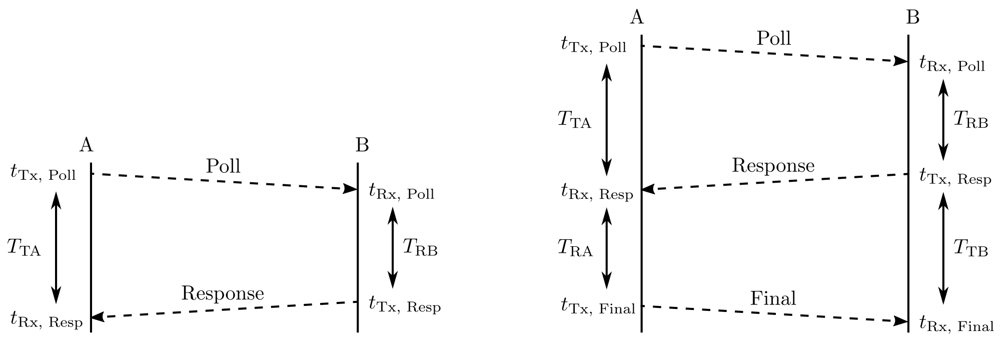
$$t_{\mathrm{Rx}} = t_{\mathrm{Tx}} + \frac{d}{c} + b\,, \tag{1.4}$$

where d is the distance between tag and anchor, c is the signal propagation speed in the environment (assumed constant or with negligible variations) and b is the bias of the time bases of the two devices.

In situations where both the tag and the anchor have their time bases synchronized, then the bias b in Equation (1.4) is equal to zero. In such case it is sufficient to send only a single message to get a distance measurement. Then the distance would be the difference of reception and transmission times multiplied by the propagation speed.

Table 1.1: Comparison of selected localization principles

		TWR	**T2A TDoA**	**A2T TDoA**
Clock synchronization		None	Anchors	Anchors
Computational demands	Anchor	Low	Low	High
	Tag	Low	Medium	Medium
Tag power consumption		Low	Low	High
Simultaneous users		Tens	Hundreds	Unlimited



(a) Single sided (b) Double sided

Figure 1.6: TWR messaging schemes

However, having both anchors and tags synchronized is not a suitable option. Mainly because it raises the tag's power usage (thus reducing battery life) and, more importantly, because the actual distance between the devices must be known for a precise time base synchronization. Of course, for mobile devices such as the tag the distance is not available prior to its localization.

Consequently, the bias must be eliminated using a different approach. The *Two Way Ranging* protocol uses a series of messages sent between the anchor and the tag to eliminate bias and determine distance.

There exists a number of TWR versions that differ in terms of number of messages exchanged, duration of a single measurement, computational demands, accuracy and sensitivity to bias drift [12]. We will mention three variants in total, the Single Sided TWR and the Symmetric and Asymmetric variants of the Double Sided (SDS and ADS) TWR, where only the latter two are of a practical use [12].

The error terms specific to the each of the TWR versions will be shown without a detailed derivation, as the TWR is not the main focus of this work. The derivation is available in [12].

The simplest form of TWR is the Single Sided TWR, where only two messages are sent. We will denote them as poll and response message. The message exchange is depicted in Figure 1.6a.

The measurement is initiated by device A sending the poll message. Upon the poll reception, B device sends response. For each message the transmission and reception times are stored and combined into two delays, the round trip delay T_{TA} on the A's side and the reply delay T_{RB} on the B's side [12].

$$T_{\text{TA}} = t_{\text{Rx,Resp}} - t_{\text{Tx, Poll}}\,, \quad T_{\text{RB}} = t_{\text{Tx,Resp}} - t_{\text{Rx, Poll}}\,. \tag{1.5}$$

Using these two delays we are able to eliminate the mutual time base bias and express the measured[5] *time of flight* $\hat{\tau}$ as a bias-free difference of round trip delay and reply delay. We get the distance by multiplying the $\hat{\tau}$ with the speed of light c.

$$\hat{d} = c \cdot \hat{\tau} = c \cdot \frac{1}{2}(T_{\text{TA}} - T_{\text{RB}}) \tag{1.6}$$

However, the real UWB transceivers suffer from the clock drift ($\dot{b}$), which results in prolonging or shortening of the delays T_{TA} and T_{RB} according to the drift of device A or B,

[5]Variables with the $\hat{}$ symbol above have the meaning of estimate/measurement of the real value affected by the measurement errors (e.g. $\hat{t}$ being the measurement of t).

respectively. If we assume that the clock drift is constant during the message exchange we can express the error resulting from the clock drift as (full derivation available in [12])

$$\hat{d} - d = c \cdot [\delta f_a \tau + \frac{T_{\mathrm{RB}}}{2}(\delta f_a - \delta f_b)] \,, \tag{1.7}$$

where d is the true distance, τ is the time of flight and δf_a and δf_b are the clock drifts (relative deviation from the nominal frequency) of A's and B's clock, respectively.

The UWB devices have the clock drift δf specified by the standard IEEE 802.15.4 [2] to be within the range ± 20 ppm. Using the expected values of the delays (τ in tens of nanoseconds, T_{R} in units or tens of milliseconds) and the worst case values for the clock drift the value of the ranging error easily exceeds several meters [12]. For this reason the single sided TWR is hardly usable for positioning.

The Symmetric Double Sided SDS-TWR reduces the error caused by the clock drifts by adding a *final* message to the messaging scheme as can be seen in Figure 1.6b. This message is sent by the device A after receiving the response message.

By adding an additional message we can calculate another round trip delay and reply delay. The distance $\hat{d}$ is then calculated as follows

$$\hat{d} = c \cdot \hat{\tau} = \frac{c}{4}(T_{\mathrm{TA}} - T_{\mathrm{RB}} + T_{\mathrm{TB}} - T_{\mathrm{RA}}) \,. \tag{1.8}$$

The T_{TA} and T_{RB} are the same delays as in the Equation (1.6) and the T_{TB} and T_{RA} are the round trip and reply delays associated with the response and final messages.

Following similar procedure of introducing the clock drifts into the distance equation the ranging error has the following form [12]

$$\hat{d} - d = \frac{c}{4}[2\tau(\delta f_a + \delta f_b) + (\delta f_a - \delta f_b) \cdot (T_{RB} - T_{RA})] \,. \tag{1.9}$$

We can observe that the error does not depend on the absolute reply delays anymore. Instead it is proportional to their difference. Consequently, the ranging error due to the bias drift is mitigated if the reply delays are equal. In practice securing the equality of the reply delays can be difficult due to the imperfections in message transmission timing. Using the typical values for the variables in the Equation (1.9) we get that the clock drift error is reduced to reasonable levels of several centimeters making the clock drift no longer a dominant error source [12].

While SDS-TWR greatly reduces the error caused by the clock drift, it heavily depends on the ability of the devices to keep the reply delays as similar as possible. Scheduling schemes that take the symmetry requirement into account were already introduced [13]. However, utilization of the *Asymmetric Double Sided* TWR removes the necessity of symmetric communication.

The ADS-TWR is almost identical to the SDS-TWR up to the point where the distance is calculated [12]. The range estimate is no longer a simple linear combination and takes the following form:

$$\hat{d} = c \cdot \hat{\tau} = \frac{T_{\mathrm{TA}} T_{\mathrm{TB}} - T_{\mathrm{RA}} T_{\mathrm{RB}}}{T_{\mathrm{TA}} + T_{\mathrm{RA}} + T_{\mathrm{TB}} + T_{\mathrm{RA}}} \,. \tag{1.10}$$

It has been proven that the nonlinear nature the ADS-TWR does not cause any problems and its performance in the terms of estimate variance is similar to the SDS-TWR [14].

The error for ADS-TWR is equal to

$$\hat{d} - d = c\frac{\delta f_a + \delta f_b}{2}\tau. \tag{1.11}$$

In comparison, the error achieved by the ADS-TWR is same as the error for SDS-TWR error when the reply delays are equal. However the ADS-TWR is totally independent of the reply delays meaning the error due to clock drift is negligible under the assumption that it is constant during the measurement [12].

As we have seen in this section, the SDS and ADS-TWR achieve superior performance in terms of clock drift induced errors when compared to the Single Sided TWR. For the ADS-TWR the error does not depend on the reply delay difference, unlike for the SDS variant. However the independence of the ADS error on the delays is bought by the increased computational cost on the distance calculation (involves a division by a variable).

For summary, the TWR provides a method of measuring distances without a mutual clock synchronization. Nonetheless, the necessity of bi-directional communication results in lower achievable measurement frequency and higher power consumption.

1.2.2 Time Difference of Arrival

When using ToF or ToA principle, the position is estimated as the intersection of several circles or spheres that were determined by the measured distance between the tag and each anchor. Additionally, the ToA measurements also contain bias between the time base of the measuring anchor and the reference time base, if the anchors are not synchronized. Before the estimation, any bias present has to be removed from the measurements, for example by subtracting the measurements in cases when the bias is equal for all of them. The resulting quantity is called the *Time Difference of Arrival* (TDoA) and can be multiplied by the propagation speed to obtain the difference of distances.

The *Time Difference of Arrival* principle uses the difference of two distances for the position estimation, rather than the absolute distance. A single TDoA measurement $h_{i,j}$ is obtained by subtracting two ToA measurements (related to the reception of the same tag message) from two different anchors i and j. For the difference of distances it holds

$$h_{i,j} = d_i - d_j = \|\boldsymbol{r}_i - \boldsymbol{r}\| - \|\boldsymbol{r}_j - \boldsymbol{r}\| \ . \tag{1.12}$$

In the equation above the vectors $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ are the position vectors of anchors i and j, respectively, and the $\boldsymbol{r}$ is the position of the tag. The Equation (1.12) restricts the set of the possible tag positions to a hyperbola (in two dimensions) with focal points at anchors i and j. For this reason the positioning systems using TDoA principle are sometimes referred to as the *hyperbolic positioning* systems.

At least three independent TDoA measurements are required (four ToA measurements) to get an unambiguous positioning solution in two dimensions. In three dimensions the minimal measurement count is also three TDoAs measurements but only if the half-space is specified. The estimation of the tag's position is then graphically equivalent to the finding of the intersection point of all measured hyperbolae (two dimension) or hyperboloids (three dimensions).

The presented principles and solutions work well in three-dimensional positioning, nevertheless, for simplicity we will consider only the two-dimensional case with hyperbolae.

Figure 1.7 contains a plot of two hyperbolae and a localized tag at the point of their intersection. The hyperbolae were determined by combining the measurements from anchors 1 and 3 and 2 and 3.

The TDoA in UWB networks exists in at least two variants that are the *Tag to Anchor* (T2A) and *Anchor to Tag* (A2T). Even if both variants estimate position using similar tech-

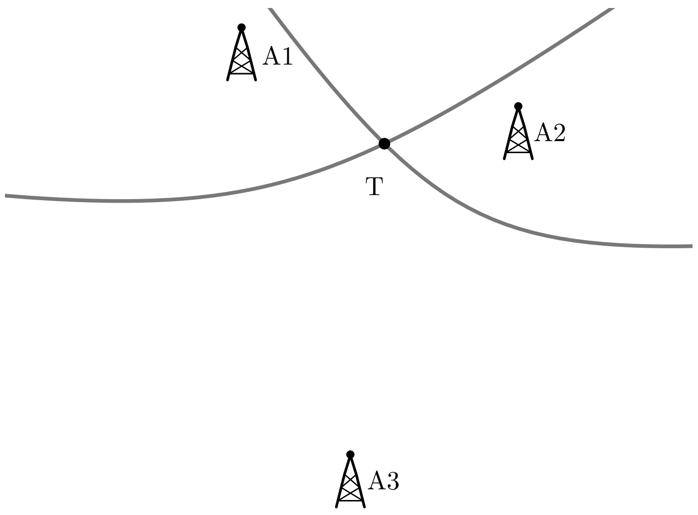


Figure 1.7: Time Difference of Arrival positioning principle

niques (intersection of hyperbolae), they are quite different. They mostly differ in the messaging schemes and the processing of the measurements.

We start by discussing how the measurements are done and what are the pros and cons of both methods. Next we will outline the measurement processing for both variants. For both variants we assume that the time bases of the anchors are synchronized.

In the T2A-TDoA the tag broadcasts a *blink* message that is received by the anchors. The anchors then send the measured times of arrival to the computation node for the position estimation.

With the A2T-TDoA the roles switch as the anchors periodically broadcast localization messages that are received by the tag. The tag is able to estimate its own position with the received data, provided that the anchor positions are known.

From this short description we can see with the T2A-TDoA, tag has to transmit only a single message to get enough data for estimation of its position. Also, there is no need for tag-anchor synchronization as the estimation is done purely from the ToA measurements. This greatly increases the battery life of the tag as the message transmission has far lower power usage than the reception and that no demanding calculations on the tag's side are required.

However, the positions have to be estimated at the computational node of the network, making them unavailable to the users if they are not transmitted back to them. Also, while the maximal number of simultaneously operational tags is much higher than with TWR, there still exists the upper limit for the position fix frequency (which is in order of thousands) that is determined by the air utilization, the amount of network control messages and by the speed of calculations of the position estimates.

This limit is seemingly removed with the *Anchor to Tag* TDoA scheme, where the position estimates are calculated by the users themselves. Using the reception times and known transmission times, the tag is able to calculate TDoAs and estimate its own position. Consequently, the system is able to support an arbitrary number of simultaneously active users, i.e. the situation is remotely similar to the Global navigational satellite systems (GNSS), but at the cost of increased tag's power demands due to the frequent reception and calculations. Moreover, A2T-TDoA has to deal with difficulties connected with the bias drift of the tag, which is solved non-trivially. The reason will reveal itself once we describe the differences of how the TDoA variants process measurements.

If we express the distances d_i using the times of arrival and transmission (see Equation (1.4)) and combine it with the Equation (1.12), we get the following expression

$$h_{i,j} = c \cdot \left[(t_{\mathrm{Rx},i} - t_{\mathrm{Rx},j}) - (t_{\mathrm{Tx},i} - t_{\mathrm{Tx},j}) - (b_i - b_j) \right], \tag{1.13}$$

where the b_i and b_j are the clock biases of the tag time base to the time base of anchor i and j, respectively.

As we have stated in Section 1.2.1 the bias presence in the measurements renders the measurements unusable for the positioning and therefore it has to be removed. Unlike the TWR, the TDoA principle tries to remove the bias by synchronizing the time bases of the anchors. The time synchronization of the anchors has to suffice as it is unfeasible to have tags synchronized with anchors (as stated in the Section 1.2.1).

At this point we have to distinguish the T2A and A2T variants of the TDoA as the exact form of the measurements $h_{i,j}$ differ for each variant.

Firstly we will discuss the *Tag to Anchor* measurements. In this case each tag broadcasts a *blink* message to the surrounding anchors. Therefore each of the transmission times $t_{\mathrm{Tx},i}$ is equal $t_{\mathrm{Tx},i} = t_{\mathrm{Tx},j}$ for every i and j and in the measurement (1.13) they cancel out. The time bases of the anchors are synchronized so there is no mutual bias between any two anchors. Also the *blink* message is received almost simultaneously (with gaps long units of microseconds at maximum) by the anchors, which means that the b_i remains almost constant, because the error caused by the bias drift remains negligible (approximately units of picoseconds). Thus we can consider the tag biases b_i equal and that in the measurements they cancel out as well. From this we can see that for T2A-TDoA it is sufficient to have only anchors synchronized and that the measurements $h_{i,j}$ can be directly used for position estimation.

For the A2T-TDoA the bias elimination is much more difficult. In this variant the anchors broadcast localization messages that are received by tags. However, the UWB was not developed for simultaneous reception of multiple messages [2]. The anchors then have to broadcast the messages with some delay between each transmission to give tags enough time to process the received message and prepare for the next reception. The transmission times can be embedded into the data parts of the localization messages so it is not problematic to compensate for the delay in the measurement. But due to the transmission spacing the error induced by the bias drift is no longer negligible. Even when the anchors are synchronized, the tag is still not and the biases are no longer equal, i.e. $b_i \neq b_j$ for anchors i and j.

The transmission delays have to be long enough for the give tags enough time for processing. At the same time the delay must be as short as possible to minimize drift-related errors. We can expect that the transmission delays will be at least as long as it takes to send a single message, but possibly longer to include time margin for the processing. The expected length of the delays is in order of units or tens of milliseconds.

The error (bias difference) induced by the bias dynamics can be written as

$$b_i - b_j = \dot{b} \cdot (t_{\mathrm{Tx},i} - t_{\mathrm{Tx},j}). \tag{1.14}$$

Again, if we consider the acceptable range for the bias drift $\pm 20\,\mathrm{ppm}$, specified by the standard IEEE 802.15.4 [2], we can calculate the expected error. With the expected transmission delays of several milliseconds the bias drift error easily gets into range of tens of nanoseconds which again makes the measurements unusable.

The bias change is significant and it is necessary to compensate for it in order to enable the tag's localization. Probably due to the bias difficulties there are not many A2T-TDoA solutions available for the UWB networks. Therefore, solving this problem and making A2T-TDoA localization possible is the main focus of this thesis and we will discuss this topic in more detail and propose a solution in Chapter 3.

2 Synchronization and Tag to Anchor TDoA

In the context of the indoor positioning and navigation, the UWB based networks have gained an increasing attention over the past decade, mostly for their promising achievable precision. In such networks the most widely used localization principles are the TWR and TDoA.

For a practical use, the TDoA is a preferred choice since it is more scalable and able to handle hundreds of concurrent users. The networks that use TWR as their core principle are suitable only for small networks with very few users (total user count usually not exceeding ten). Their use can be found for example in presentations of the UWB positioning, proofs of a localization concept, proximity detection or during production tests and calibration.

In the previous chapter we have introduced both the TWR and TDoA localization principles, their pros and cons and outlined the implementation requirements and difficulties. For the TDoA we have defined two messaging schemes that we will consider, the *Tag to Anchor* and the *Anchor to Tag*. In the UWB-TDoA networks the T2A variant is used significantly more often than the A2T one. The probable reason is that the A2T-TDoA measurements are heavily affected by the drift of tag's clock. The process of the clock drift compensation is not trivial, as will be shown in Chapter 3.

Here we will be focusing on the T2A-TDoA. In this variant, a tag sends a *blink* message that is received by all the anchors within the signal range. Every anchor stores the time of arrival of the blink message. The measured time together with the tag's identification are then sent from each anchor to the computation node where the position is estimated.

The implementation of an UWB-TDoA system or network can be broken down into three main problems that are summarized below. The designer may choose to solve them separately, however, that can prove to be difficult, as each problem may affect the others.

- *Anchor synchronization*, usually done through an UWB channel using a single anchor as a time reference. The selected anchor periodically sends synchronization messages that the other anchors use for their own synchronization [3, 13, 15].

  There also exists a protocol that does not require synchronization of the anchors. Such protocol is proposed by [16], where the biases are eliminated using relations between successive receptions of tag's messages. However, such methods have certain drawbacks, for example relying on inter-message relations that do not have to be present. Another drawback is the introduction of additional communication load to the UWB channel, which is higher than with synchronization only.

- *Positioning data collection and position estimation*; there has to be established a communication path between the anchors and the computation node that is used for sending positioning data (e.g., time stamps).

  The proper algorithm for the position estimation must be chosen as well. As the problem is nonlinear, a popular choice is an *Extended Kalman Filter* (EKF) or any of the Newton-like methods, which solve the problem as a *nonlinear least-squares* (NLSQ) problem [17, 18] (e.g. Levenberg-Marquardt method).

- *Scalability*; the system must be scalable in order to cover large areas, such as storage halls. The scalability mainly depends on the capabilities of the deployed synchronization algorithms and the realization of power and data interconnections.

  One of the synchronization scaling approaches is to have many separate anchor subnetworks (usually denoted as synchronization domains) with a reference anchor in each domain [19]. The problems arise on the boundaries of the domains where we get time stamps related to a single tag message but measured with respect to different time bases. Consequently, such TDoA measurement is useless.

  A different approach is to spread the synchronization packets from single reference anchor through out the network with the use of so-called *relay* anchors. By chaining the otherwise separate domains together we are able to have the whole network in single synchronization domain [10].

From the three problems listed above, two problems, synchronization and estimation, were solved in our previous works [3, 4]. The scalability was solved in [10], where we have proposed the Chained synchronization algorithm.

This chapter gives a summary of the synchronization and estimation solution together with a description of the Chained synchronization in a later section.

Since this work's primary focus is not on the T2A-TDoA and anchor synchronization topics, we will provide the descriptions without discussing the results. Instead we will provide illustrative results in Appendix B.

2.1 Synchronization

The synchronization of the anchor time bases plays a crucial role in the TDoA localization. Without it, the measured receive times would be unusable as every time would be measured with respect to a different time base and without knowledge of the mutual bias of these time bases. In order to remedy this, the anchor time bases have to be precisely synchronized, usually with sub-nanosecond accuracy.

The purpose of synchronization is to bring the mutual time bias of any two anchors within the network to zero, i.e., make every anchor to measure time with respect to a single common time base or time domain. Thus, a reference time domain has to be chosen. The specific choice of the reference domain is not important. A straightforward and valid option is to select a single anchor within the network to which the others synchronize. We will label the time-reference anchor as the master anchor (having its master domain) and the rest as slave anchors.

The synchronization should introduce minimal additional load to the communication channel, since it is also used for the positioning data (e.g., blink messages). Thus, one-way synchronization methods are preferred over the multi-way ones (such surely send more than one message). Also, using a one-way synchronization we make the protocol easier, since such method can broadcast the messages (send message to all devices that are able to receive), and do so without implementing a message delivery checks. In one-way synchronization the master anchor periodically broadcasts synchronization messages that contain the time of the message transmission expressed in the master time domain.

Synchronization can be done through a wired channel or through wireless channel, but since we are interested in easily scalable solutions we chose the wireless option.

For the synchronization we will use the second-order linear approximation of the bias (1.3), where b_i expresses the time domain bias of ith anchor to the master domain

$$b_i[k] = b_i[k-1] + \dot{b}_i[k-1]T_k + \frac{1}{2}\ddot{b}_i[k-1]T_k^2 \,, \tag{2.1}$$

where T_k represents the elapsed time between epochs k and $k-1$. The term measurement epoch, or simply an epoch, refers to a time instance between two measurements or between two position fixes in the positioning context.

A comprehensive overview and comparison of the wireless one-way synchronization methods is offered by [15]. The article compares five methods with varying complexity, with every method using only a first-order linear model of the bias (with only the bias b_i and its drift $\dot{b}_i$). The methods range from a simple linear interpolation, over a PID controller to a *Kalman filter*. Each method assumes that it periodically receives synchronization messages containing master time. We will refer to the period of the synchronization messages as the synchronization period T_k. It is shown by [15] that the performance of a method depends also on the length of synchronization period, which can then be also viewed as a design parameter for the system.

The simplest of the methods is the linear interpolation. In each synchronization period the current bias is calculated

$$b_i[k] = t^A_{\text{Rx, sync}}[k] - t^M_{\text{Tx, sync}}[k], \tag{2.2}$$

where the superscripts denote in which time domain the value was measured, the A is for slave anchor domain and M for the reference time domain. Reception times of a blink messages are stored until the next synchronization message arrives. Then the time stamp correction for each blink is calculated using linear interpolation. Consequently, the TDoA measurement formation and position estimation is inherently lagged.

Another methods proposed by the [15] are the proportional-integral (PI), the proportional-integral-differential (PID) and the proportional-integral-integral (PII) control loops. The three methods are very similar, however, they differ notably in terms of performance and stability, for more details consult [15].

The PI-based control loops take a different approach of estimating the correction value. Instead of estimating the bias value between the slave and master time domains, the loops try to estimate the receive time $\hat{t}^A_{\text{Rx}}[k]$ of the next synchronization message. When the synchronization message arrives, the master transmit time and the reception time of the message is passed as inputs to the control loop. The difference between the actual $t^A_{\text{Rx}}[k]$ and estimated $\hat{t}^A_{\text{Rx}}[k]$ reception time of the message is used to drive the PI-based control loop. The correction value for the receive times of the blink messages can be calculated right after the message reception with the use of the estimated reception of the synchronization message $\hat{t}^A_{\text{Rx}}[k]$ and the transmission time of synchronization message $t^M_{\text{Tx}}[k]$. The detailed description with schematics can be found in [15].

The final presented method is the discrete Kalman filter.[6] The estimated state vector comprises of the bias b and the bias drift $\dot{b}$. The filter has no control input and accepts the direct measurement of the bias, which is the difference of the time of reception and transmission of the synchronization message.[7] The predicted value of the bias is used for the correction of the blink reception time. The predicted bias is obtained by performing the *time update* step of the KF.

[6]We will refer to a discrete version of Kalman filter simply as a Kalman filter.

[7]This is not entirely true as the measurement is missing the propagation delay between master and slave anchor. We will return to this topic in Section 2.1.1.

The discussed synchronization methods were compared using the achieved localization accuracy for several values of the synchronization period ranging from 150 ms to 900 ms. The best results were achieved using the Kalman filter that outperformed the rest even for longer synchronization periods [15]. Even though the Kalman filter is the most complex among the methods, its stability, performance, variance estimates and precise predictions easily outweigh the increase of computational demands.

Based on these reasons we have chosen to use the Kalman filter proposed by the [15] as a basis for one-way synchronization of the UWB network. We have added the second-order state to the filter, the bias drift rate $\ddot{b}$, in order to further improve its performance, especially during the warm-up phases of the devices.

In the following subsections we will describe the implementation of the Kalman filter for one-way point to point (master to slave) synchronization. We will also describe the Chained synchronization extension, which allows to synchronize widespread networks by relaying the synchronization messages.

2.1.1 Point-to-Point synchronization

In the previous text we have briefly discussed the possible algorithms for the anchor synchronization presented by the [15]. From the listed algorithms the Kalman filter stood as the one achieving the best performance. We have implemented the KF for the purpose of synchronization in our previous works [3, 4].

The Kalman filter is a stochastic state estimator of a system, where we are unable to measure the states directly. However we are able to observe the internal state by measuring the system output. The measurement is typically affected by the additive white Gaussian noise [20, 21].

The Kalman filter is a two step recursive algorithm where the state development is predicted in the *time update* step and corrected afterwards when the current measurement is included during the *measurement update* step. In its original form, the Kalman filter is only used on linear systems. For such systems the Kalman filter is optimal in a sense that it minimizes the mean squared estimation error [21].

The filter recursively estimates the system state vector $\boldsymbol{x}$ that evolves in time according to the linear system state model

$$\boldsymbol{x}[k] = \mathbf{F}\boldsymbol{x}[k-1] + \mathbf{B}\boldsymbol{u}[k-1] + \boldsymbol{w}[k]\,, \tag{2.3}$$

$$\boldsymbol{y}[k] = \mathbf{H}\boldsymbol{x}[k] + \boldsymbol{v}[k]\,, \tag{2.4}$$

where $\mathbf{F}$ is the system matrix, $\mathbf{B}$ is the control matrix, $\boldsymbol{u}$ is control input, $\mathbf{H}$ is output matrix (measurement model) and $\boldsymbol{y}$ is the system output. In general, the matrices are time variant as their elements may change between iterations. For simplicity, however, we assume them to be time invariant.[8] The $\boldsymbol{w}$ and $\boldsymbol{v}$ are the process noise and measurement noise, respectively. Both are assumed to have the normal distribution with zero mean and covariance matrices $\mathbf{Q}$ and $\mathbf{R}$.

$$\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})\,, \quad \boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{2.5}$$

It is also assumed that the two noise vectors $\boldsymbol{w}$ and $\boldsymbol{v}$ are uncorrelated, meaning they are independent of each other $\mathrm{E}[\boldsymbol{w} \cdot \boldsymbol{v}^{\mathrm{T}}] = \mathbf{0}$ (their covariance is zero).

[8]With the matrix $\mathbf{F}$ as an exception, because we will change the elements containing the time period.

Both matrices $\mathbf{Q}$ and $\mathbf{R}$ are used as tuning parameters of the filter, since their real values might not be known or are difficult to estimate. This is especially true for the process covariance $\mathbf{Q}$. Most real-world system are non-linear, but they can be approximated on short time intervals by linear models. The approximation error is kept low by sufficient density of sampling.

Finding the real values for the measurement covariance is fortunately not that tricky. Values of the matrix $\mathbf{R}$ are usually obtained from the manual of the device or measured in a controlled environment.

Together with the state estimate the Kalman filter also keeps track of the uncertainty of the estimate. This uncertainty is represented by the state covariance matrix $\mathbf{P}$ and is updated within each time and measurement update, along with the state estimate.

Another important parameter of the Kalman filter is the estimation period, which determines how often the time update step is performed. Usually, this period is equal to the sampling period of the measurements, but there are cases in which we need the filter to predict system state in regular intervals, independently on the measurements. The possibility to do so is a great advantage of the Kalman filter.

The *time update* step predicts a new value for the state vector from the previous estimate using the system model provided. Such estimate is called *a priori* estimate, because it uses an information obtained in the past, and it is commonly labeled with a $-$ sign in the superscript. The time update equations [21] for the state estimate $\boldsymbol{x}$ and the estimate covariance $\mathbf{P}$ are

$$\boldsymbol{x}^-[k] = \mathbf{F}\boldsymbol{x}^+[k-1]\,, \tag{2.6}$$

$$\mathbf{P}^-[k] = \mathbf{F}\mathbf{P}^+[k-1]\mathbf{F}^\mathrm{T} + \mathbf{Q}\,. \tag{2.7}$$

We can see that during each time update the covariance $\mathbf{P}$ is increased by the process noise covariance $\mathbf{Q}$, by which the filter takes into account the possible imperfections of the model.

During the *measurement update* the measurement is used to correct the predicted values and compute *a posteriori* estimate $\boldsymbol{x}^+$ and covariance $\mathbf{P}^+$. In the measurement update the difference between the measurement and estimated measurement (estimated system output) is determined; this difference is called the innovation $\boldsymbol{\nu}$.

$$\boldsymbol{\nu}[k] = \boldsymbol{z}[k] - \mathbf{H}\boldsymbol{x}^-[k] \tag{2.8}$$

$$\mathbf{S}[k] = \mathbf{H}\mathbf{P}^-[k]\mathbf{H}^\mathrm{T} + \mathbf{R} \tag{2.9}$$

Kalman filter also computes the covariance matrix $\mathbf{S}$ of the innovation. By examining the elements of $\mathbf{S}$ we can detect whether the Kalman filter estimates converge to the internal state of the observed system. That is when covariance $\mathbf{S}$ is changing slightly between iterations or not at all.

Continuing with the description of the KF. The *a posteriori* estimate and covariance is calculated according to [21] as

$$\boldsymbol{x}^+[k] = \boldsymbol{x}^-[k] + \mathbf{K}[k]\boldsymbol{\nu}[k]\,, \tag{2.10}$$

$$\mathbf{P}^+[k] = (\mathbf{I} - \mathbf{K}[k]\mathbf{H})\mathbf{P}^-[k](\mathbf{I} - \mathbf{K}[k]\mathbf{H})^\mathrm{T} + \mathbf{K}[k]\mathbf{R}\mathbf{K}^\mathrm{T}[k]\,, \tag{2.11}$$

where $\mathbf{K}$ is the Kalman gain

$$\mathbf{K}[k] = \mathbf{P}^-[k]\mathbf{H}^\mathrm{T}\mathbf{S}^{-1}[k]\,. \tag{2.12}$$

The *a posteriori* estimate and covariance are then used in the next run of the Kalman filter as the previous estimated values.

When Kalman filter converges to a steady state, the time development of the innovation should look like a zero mean white noise with a covariance $\mathbf{S}$. If there is a disagreement between innovations and covariance $\mathbf{S}$, it should be understood as an indication that the filter parameters are not set correctly and further tuning or change of the model is needed.

After the introduction of the Kalman filter, we can move on to the description of its application for the purpose of the anchor synchronization. The particular implementation in [15] used a bias system model in which the state vector consists of the bias b and the bias drift $\dot{b}$,

$$\boldsymbol{x} = \begin{bmatrix} b & \dot{b} \end{bmatrix}^{\mathrm{T}}, \quad \mathbf{F} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}, \tag{2.13}$$

where T_s is the synchronization period.

The system model assumes that the bias drift is constant or that it is changing only within the process noise given by the matrix $\mathbf{Q}$. While this is mostly true for the clock in steady state (when the anchor network is running for some time), it is certainly not true for the warm-up phases of the devices after power-on. From our experiments (see [3, 4] and Appendix B) we have learned that the bias drift $\dot{b}$ is dependent on the DW1000 (its internal clock electronics) temperature and that the clock drift $\dot{b}$ changes quite rapidly.

When the drift changes considerably, due to its non-zero drift rate, the filter is not be able to keep up with the changes and the predicted bias will be repeatedly in disagreement with the measurements. As a result the innovation will no longer be zero-mean and the estimates will contain additional offsets.

To evade this problems we have added a third state that describes the *bias drift rate* $\ddot{b}$ into our implementation.

$$\boldsymbol{x} = \begin{bmatrix} b & \dot{b} & \ddot{b} \end{bmatrix}^{\mathrm{T}}, \quad \mathbf{F} = \begin{bmatrix} 1 & T_s & \frac{1}{2}T_s^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

The measurement of the bias is obtained from each of the received synchronization message by solving the eq. (1.4) for the bias,

$$b = t^S_{\mathrm{Rx},s} - t^M_{\mathrm{Tx},s}[k] - \tau\,, \tag{2.15}$$

where $t^S_{\mathrm{Rx},s}$ and $t^M_{\mathrm{Tx},s}$ is the time of reception and transmission of the synchronization messages, respectively, measured in the corresponding slave (S) and master (M) time domains. The τ is the time of signal propagation between the slave anchor and master anchor. For the correct function of the synchronization, the propagation delay τ has to be known in advance and measured as accurately as possible. Any error in the delay τ (or distance $d = c \cdot \tau$) measurements will result in offsetting the slave anchor domain from the master domain.

From the (2.15) we are able to form the measurement model $\mathbf{H}$ of the system and construct the innovation equation.

$$\nu[k] = (t_{\mathrm{Rx},s}[k] - t_{\mathrm{Tx},s}[k] - \tau) - \mathbf{H}\boldsymbol{x}^{-}[k]\,, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{2.16}$$

As for the matrix $\mathbf{Q}$ and (in this case) scalar variance R, determining the values is again easier for R as it is provided by the manufacturer. Variance R expresses the uncertainty of the time-stamping process, which includes time-stamping of the transmitted and received synchronization message. The variance for both types of time stamps is equal and its value is $\sigma_t^2 = (150\,\mathrm{ps})^2$ [15]. Therefore, the value of measurement covariance is

$$R = 2\sigma_t^2 = 2 \cdot (150\,\mathrm{ps})^2\,. \tag{2.17}$$

This value is valid for high SNR signals. For signals with low SNR the measurement variance will be larger, but typically the received signals have high SNR.

Finding the suitable values for matrix $\mathbf{Q}$ can be a tedious process and as it requires to be fine tuned until satisfactory performance is achieved. The synchronization intervals T_k may not be fixed, either due to some delay in transmission or due to a missed reception of a synchronization message. Therefore, we define the process covariance per second $\bar{\mathbf{Q}}$. To get the actual covariance $\mathbf{Q}$ during the estimation, we simply multiply the normalized covariance $\bar{\mathbf{Q}}$ with a time interval.

$$\mathbf{Q}(T) = T\,\bar{\mathbf{Q}} \tag{2.18}$$

To conclude this section we state how to correct the receive times of the blink messages. Suppose that when a blink message arrives at the anchor antenna, the time-stamping process assigns it a time stamp $t^S_{\mathrm{Rx},b}$.

Correction $\hat{b}$ of the time stamp is calculated by performing a time step for the bias value.

$$\hat{b} = \mathbf{F}_1 \cdot \boldsymbol{x}[k] = b[k] + T_b\,\dot{b}[k] + \frac{1}{2}T_b^2\ddot{b}[k], \quad T_b = t^S_{\mathrm{Rx},b} - t^S_{\mathrm{Rx},s}[k]\,, \tag{2.19}$$

where $\mathbf{F}_1$ is the first row of system matrix $\mathbf{F}$ and T_b is the delay between reception of the synchronization message and blink message. The blink ToA is then converted from the slave domain to the master domain.

$$t^M_{\mathrm{Rx},b} = t^S_{\mathrm{Rx},b} - \hat{b} \tag{2.20}$$

The Kalman filter also provides us with the variance of the correction and thus the variance of the bias estimate σ_b^2

$$\sigma_b^2 = \mathbf{F}_1\mathbf{P}[k]\mathbf{F}_1^{\mathrm{T}} + \mathbf{Q}_{11}\,, \quad \mathbf{Q}_{11} = T_b\bar{\mathbf{Q}}_{11}\,. \tag{2.21}$$

Finally, with the variance of the estimated bias, at the time of blink reception, we are able to express the total variance of the blink receive time $\sigma^2_{\mathrm{Rx},M}$ in master domain

$$\sigma^2_{\mathrm{Rx},M} = \sigma_t^2 + \sigma_b^2\,, \tag{2.22}$$

where σ_t^2 is the variance of the time stamp measurement.

The calculated variance can be used to asses the measurement quality, which can help to evaluate which measurements are better suited for the position estimation.

2.1.2 Chained synchronization

The synchronization process presented in the previous subsection assumes that there is always an unobstructed line of sight (LOS) between a slave and the master anchor. This poses a limit on the UWB network size and the area where it is able to operate.

Assuring the line of sight between a single master and all the slave anchors may not be possible. For example consider a situation, where we have to provide localization inside of a building across multiple rooms. Surely we are able to guarantee a LOS within each room but not across all of them.

It is possible to synchronize slave anchors even without a line of sight, but we have to keep in mind that any obstruction in the signal path adds an unknown delay to the signal time of flight. This delay creates an additional offset to the input of the synchronization KF that disrupts the synchronization. Moreover, each obstruction also attenuates the signal, effectively shortening the range of synchronization signals.

A possible way to partially solve this issue is to break the network to several isolated synchronization domains, each with its own master anchor, with no inter-group communication allowed. Every slave anchor within a domain has a LOS to the domain's master anchor. While this solution is fairly simple to implement, it is usable only if the movement of users is restricted to a single group only. Once any user ventures on the boundary of two domains then it may not be possible to successfully localize him as the time measurements will come from two different time domains with unknown mutual bias. The Chained synchronization algorithm overcomes such limitations by connecting the whole UWB network into a single network-wide synchronization domain.

As already mentioned, we are often unable to assure the LOS between master and each slave anchor. However, instead of insisting on direct LOS paths for the synchronization we rather chain the anchors together to create a composite LOS path with the master anchor at the beginning and the slave anchor at the end of the chain. These chained anchors have LOS to the intermediate anchors in the chain and their role is to *relay* the synchronization messages from the master further down the chain.

Utilizing the intermediate LOS links to substitute for the direct LOS is the key idea of the Chained synchronization algorithm [10]. The algorithm introduces new anchor type into the network, the relay anchor.

The function of a relay anchor is straightforward. Upon receiving a synchronization message at time $t^R_{\mathrm{Rx},s}$ from a relay (or the master) that is previous in the chain, the relay synchronizes itself using the algorithm we described in Section 2.1.1. Relay then broadcasts a new synchronization message to the slaves (or relays) that are further in the chain at time $t^R_{\mathrm{Tx},r}$ after a defined delay T_{Tx} elapses [10]. The transmitted relay message is similar to the synchronization message sent by the master. The message includes the time of transmission from the relay $t^M_{\mathrm{Tx},r}$ converted to the master domain by the estimated bias value b_r. These messages can be received by other relays that resend them to anchors in their vicinity.

$$t^M_{\mathrm{Tx},r} = t^R_{\mathrm{Rx},s} - b_r + T_{\mathrm{Tx}} \tag{2.23}$$

In addition to the transmission time value the relay anchor also includes the variance of the transmission time, providing the measure of quality of its own synchronization and transmission time estimation. Note that this is enabled by the use of Kalman filter. The transmission variance is calculated in a similar fashion as it was for the blink variance in the (2.21) but with an additional term describing the transmission time stamp variance [10].

$$\sigma^2_{\mathrm{Tx}} = \mathbf{F}_1\mathbf{P}\mathbf{F}_1^{\mathrm{T}} + T_{\mathrm{Tx}}\bar{\mathbf{Q}}_{11} + \sigma^2_t \tag{2.24}$$

Relay anchors use the synchronization KF, which we have described in Section 2.1.1, with a modified measurement equation. In this case both the message receive $t_{\mathrm{Rx},s}$ and transmit $t_{\mathrm{Tx},s}$ times have an associated variance since the message can originate either from master or relay anchors. The variance of the receive time stamp is given by the measuring variance $\sigma^2_t = (150\,\mathrm{ps})^2$ and the transmission time variance σ^2_{Tx}, which is obtained from the received relay message. We combine both variances in the measurement variance R.

$$R = \sigma^2_t + \sigma^2_{\mathrm{Tx}} \tag{2.25}$$

Of course we also have to use an appropriate value for the propagation delay τ in the measurement equation (2.15).

The Chained synchronization algorithm provides us a way to expand our UWB network beyond a single room or hall. By introducing the relay anchors we are able to spread the synchronization messages throughout the whole network and have every anchor belong to a single time domain.

The algorithm itself has been proposed in [10] along with the predictions of the performance. The performance was then experimentally evaluated in to following article [11]. Most attention has been given to the synchronization error, which is the difference in synchronization through the direct path (from master to slave) and the relayed path.

Results of the evaluations confirmed the predictions. Considering the synchronization error, the further from the master a relay is the higher is its synchronization error. This is due to the imperfections of the individual relays, which originate either from systematic errors (such as calibration errors) or from the errors of rather random character (such as power fluctuations, effects of the environment). Each error contributes to the offset of the relay's time domain and thus to the synchronization error. Moreover, this error is cumulative and after performing several hops (passing several relays) the error exceeds several nanoseconds. This may greatly deteriorate the localization performance if the measurements originating from *distant* relay time domains are combined [11]. However we can mitigate this issue if the relay domains are physically far enough, so that their measurements are less likely to be combined.

The error can be lowered during production and network installation. In production the calibration of the equipment delays has to be done precisely. Those delays contain the antenna delay and power dependent delays [22]. Also precise anchor position determination helps to reduce the error as the ranges to master and relay anchors enter the calculations. Although critical, the process of determination and compensation of these delays is beyond the scope of this work, but it will be a subject of the future works.

By allowing the relay anchors to act as a synchronization sources for the slave anchors, the algorithm provides a possibility to create redundant synchronization paths within network. In the articles [10, 11] we have also investigated the effect of the choice of topology on synchronization error.

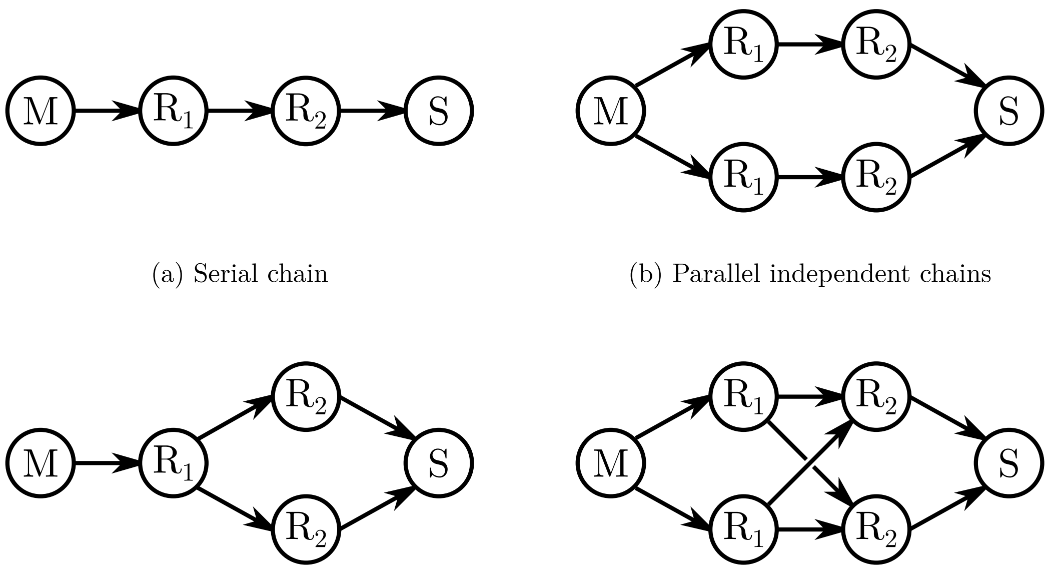
Few chain examples can be seen in Figure 2.1, starting with a simple sequential path. For the redundancy we can for instance construct multiple independent paths (Figure 2.1b) from which a slave anchor can receive synchronization. The advantage of using independent paths is having a redundant path, but also the possibility to combine the synchronization messages coming from them, which usually reduces the synchronization error [11].

We are not limited to the two mentioned topologies as we can create much complex ones where the paths join and split at several points, such as the mesh topology in Figure 2.1d.

2.2 Construction of the TDoA pairs

Synchronization of the anchors using the discussed algorithms makes TDoA localization of the tags within an UWB network possible.

As was already discussed in the previous sections, the measurements for the *Tag to Anchor* TDoA variant are created from the ToA measurements of a blink message received by the anchors. All such time stamps must be available to the position estimation algorithm, including the identification of the tag and anchor. Therefore, every anchor sends the blink time stamp,



(a) Serial chain

(b) Parallel independent chains

(c) Partially shared

(d) Mesh

Figure 2.1: Examples of chain topologies [10]

tag address, own address and additional data such as RSSI or tag's battery voltage to the computation node for the position estimation.

The computation node creates the TDoA measurements from the input anchors data. This is done by taking a pair-wise differences of the receive time stamps. We will discuss how to choose the pairs later in this section.

A TDoA measurement $\tilde{h}_{ij}$ is constructed by taking the reception times of a blink message received by the anchors i and j

$$\tilde{h}_{ij} = t_{\mathrm{Rx},i} - t_{\mathrm{Rx},j}. \tag{2.26}$$

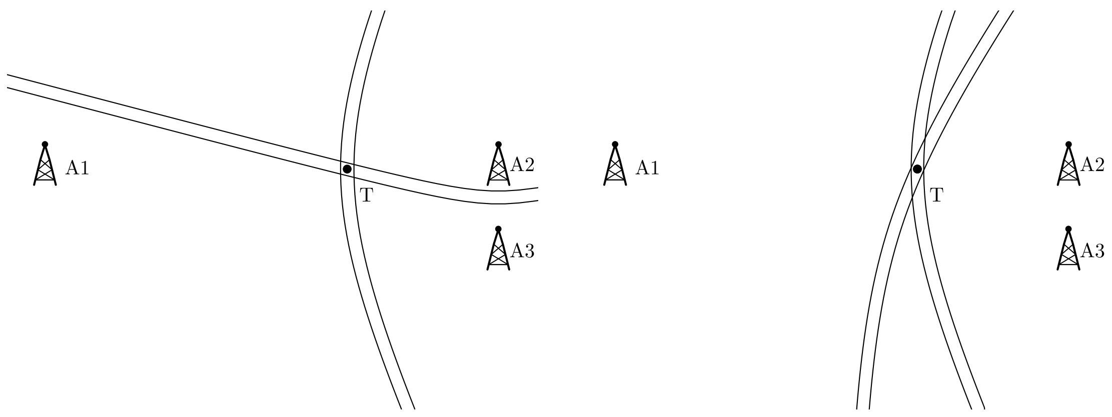
The measurements are gathered into a TDoA measurement vector $\tilde{\boldsymbol{h}}$. For convenience we can also gather the reception times into a single vector $\boldsymbol{t}_{\mathrm{Rx}}$ and describe the pairing using a matrix $\mathbf{D}$, then we can create the vector $\tilde{\boldsymbol{h}}$ using matrix multiplication.

$$\tilde{\boldsymbol{h}} = \mathbf{D}\boldsymbol{t}_{\mathrm{Rx}} \tag{2.27}$$

The matrix $\mathbf{D} \in \mathbb{R}^{m\times(m-1)}$, where m is the number of ToA measurements, is called a combination matrix. Each row of the matrix contains a single 1 and a single -1 to describe a ToA pair which forms a TDoA measurement. Following is an example structure of the $\mathbf{D}$ matrix.

$$\mathbf{D} = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \end{bmatrix} \tag{2.28}$$

If the pairs are chosen carefully, we can construct the $\mathbf{D}$ matrix such that its rows are linearly independent. Then the vector $\tilde{\boldsymbol{h}}$ contains measurements with maximal information.



(a) $\mathbf{D}_1$ results in hyperbolae with low DOP (b) $\mathbf{D}_2$ results in hyperbolae with high DOP

Figure 2.2: The effect of matrix $\mathbf{D}$ choice on DOP

Equation (2.26) expresses the difference in the blink message reception. Given the anchor time base synchronicity[9], the reception time difference is also equal to the difference of signal propagation times τ_i and τ_j from the tag to the two particular anchors.

$$t_{\mathrm{Rx},i} - t_{\mathrm{Rx},j} = \tau_i - \tau_j \tag{2.29}$$

The computed time differences are typically in the order of nanoseconds. For the sake of avoiding arithmetic difficulties during the position estimation, it is convenient to convert the times into distances and work with larger numbers. Then the equation (2.26) can be expressed using the positions of the tag $\boldsymbol{r}$, ith and jth anchor.

$$h_{ij} = \|\boldsymbol{r}_i - \boldsymbol{r}\| - \|\boldsymbol{r}_j - \boldsymbol{r}\| = c \cdot (\tau_i - \tau_j) = c \cdot (t_{\mathrm{Rx,i}} - t_{\mathrm{Rx},j}) \tag{2.30}$$

The equation above is an equation for hyperbola with parameter h_{ij} and focal points at the anchors. The hyperbola is a set of possible tag locations at the time of the tag's blink transmission.

Multiple TDoA measurements must be available in order to find the position of the tag. Each TDoA measurement corresponds to a hyperbola on which the tag lies. Consequently, the tag position estimate is at the intersection point of all measured hyperbolae.

The hyperbolae are determined by the selection of anchor pairs used for construction of the TDoA measurements and thus by the structure of the combination matrix $\mathbf{D}$.

Now let us consider an example of tag localization, where we try to estimate its position with measurements from three anchors, but with two different combination matrices $\mathbf{D}_1$ and $\mathbf{D}_2$. The example setup with the resulting hyperbolae is depicted in Figure 2.2.

$$\mathbf{D}_1 = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}, \quad \mathbf{D}_2 = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \tag{2.31}$$

In Figure 2.2 we can observe how the choice of combination matrix $\mathbf{D}$ influences the resulting hyperbolae deduced from identical set of receive time stamps. We can see that in Figure 2.2a

[9]For the purposes of the position estimation we assume that the anchors are synchronized and thus they measure w.r.t. a common time base. Therefore, for the remainder of this chapter we omit the superscripts, denoting the time domain.

that the hyperbolae intersect at an almost orthogonal angle, as opposed to the situation in Figure 2.2b.

If the measurements were perfect, then the specific choice of **D** would not affect the results of the estimation, as the hyperbolae would intersect precisely at a single point.

However the real-world measurements are always affected by error sources, which in turn cause the hyperbolae not to intersect at a single point. The hyperbolae then resemble zones, rather then lines, which is illustrated in Figure 2.2.

Instead of a point of the intersection, the zoned hyperbolae have an area of intersection with tag being anywhere within the area.[10] In Figure 2.2b the area is much larger than in Figure 2.2a, because the two hyperbolae intersect at much sharper angle, so that their arms are almost parallel.

This brings us to a parameter called *Dilution of Precision* (DOP). It represents the measure of sensitivity of the estimate on the measurement inaccuracy. It tells how much the estimated position changes if we disturb the measurements by a small value. The higher the DOP the more sensitive the estimate will be to the input date and the more it will be prone to errors.

The calculation of the DOP is a topic which deserves its own section and we devoted Appendix A for its description.

The DOP is related to the measured TDoA and consequently with the combination matrix **D**. As a result, if we modify the matrix **D** the DOP will also change.

We can see now that using a single **D** matrix for the whole estimation process may not be optimal in the terms of achievable DOP, thus localization accuracy as well. Therefore we may want to change the particular measurement pairing when the user moves in an area with high DOP. However, we do not know the DOP value prior to the position estimation. It is apparent that the task of finding the optimal **D** is not straightforward.

When the total number of anchors is low we should be able to compute the DOP (for computation see Appendix A) in advance for every possible **D** matrix. During the estimation we could choose the optimal **D** based on the pre-calculated DOP and last position estimate. However, this approach is very memory demanding for higher number of anchors.

Another approach could be such that we keep track of the last used **D** matrix and last DOP for each tag. If the new position estimate achieved higher DOP we try to permute the **D** for the next estimation epoch.

It is apparent that the choice of **D** is not direct one. To the author's knowledge there is not a published method or algorithm for the choice of the TDoA pairs or matrix **D** such that the DOP of the subsequent position estimate would be minimal (or low enough). In our future works we would like to explore this topic more and find a feasible method to optimize the **D** matrix.

2.3 Solution of the positioning equations

The last thing missing from the general description of the TDoA localization is the actual position estimation from the measured TDoA set.

Firstly, it is necessary to define the coordinate system. For the UWB positioning the preferred system is any right-handed Cartesian coordinate system, which has three perpendicular axes, labeled x, y and z. In this work we will consider the ENU system, which specifies that the x axis is pointing towards east, y towards north and z upwards.

[10]This is a fairly simplified view. In reality, there is not an area with sharp error bounds, but rather a continuous random value distribution.

From the geometric standpoint, the position is obtained by finding the intersection of the measured hyperbolae (or hyperboloids in three dimensions). For this purpose we will use the distance difference equation (2.30) from the previous section. Additionally, we will assume that during the formation of TDoA pairs a mapping $\Phi(m)$ is created that maps the measurement number m to the anchor pair i and j. The reason for that is to simplify the expressions and avoid confusion in measurement vector $\boldsymbol{h}$ having two indices in its subscripts (suggesting that it is a matrix rather than a vector). The mapping can be derived from the combination matrix $\mathbf{D}$. The measurement equation has the following form:

$$h_m = c \cdot (t_{\mathrm{Rx,i}} - t_{\mathrm{Rx},j}) \,, \quad \Phi(m) = (i, j) \,. \tag{2.32}$$

For the determination of the 3D position, three measurements h_m are sufficient. The position estimate $\boldsymbol{r}$ is then a point where all of the three hyperboloids intersect. Such point has a distance to each of the hyperbolae equal to zero, thus it satisfies the following set of equations, where the measurement is compared with the expected value.

$$h_m - (\|\boldsymbol{r}_i - \boldsymbol{r}\| - \|\boldsymbol{r}_j - \boldsymbol{r}\|) = 0 \,, \quad m \in \{1, 2, 3\} \,. \tag{2.33}$$

The measurements, however, are never perfect and are always affected by an error. As we have seen in Figure 2.2, there is no longer a single point of intersection, due to the errors (error gaps), but rather an area where the bounds overlap. Thus, more measurements are taken in order to reduce the area.

When we use more than three measurements (for the 3D case) the set of equations usually becomes over determined and has no exact solution. Instead, the position is estimated by finding a solution of (2.33) that *minimizes* the sum of squared distances to all of the measured hyperbolae. This criterion and problem of the position estimation is formulated as a nonlinear least-squares problem (NLSQ)

$$\min_{\boldsymbol{r}} f(\boldsymbol{r}) = \min_{\boldsymbol{r}} \sum_{m=1}^{M} [g_m(\boldsymbol{r})]^2, \tag{2.34}$$

where

$$g_m(\boldsymbol{r}) = h_m - (\|\boldsymbol{r}_i - \boldsymbol{r}\| - \|\boldsymbol{r}_j - \boldsymbol{r}\|) \,, \tag{2.35}$$

is the element of residual vector holding the distance to a hyperbola.

Solution can be found using the standard nonlinear optimization methods, such as gradient-descent, Gauss-Newton or Levenberg-Marquardt. In our previous works we have used the Levenberg-Marquardt method, since it provides a compromise between convergence speed and risk of divergence [3, 23]. We denote this approach as the *epoch by epoch* estimation as it is using only data corresponding to a single blink message and does not exploit the information from previous epochs.

In this work we consider an additional approach that uses also the past information and tries to estimate the position in a more continuous way. For this purpose we will use an *Extended Kalman Filter* (EKF).

In the following sections we will describe both approaches. Additionally, we will discuss the benefits and implementation of adding a soft constraint on height into the position estimation. The height constraint may help reduce the error (not only) in the vertical coordinate when the geometry of the anchors is flat and prone to errors in estimation of the tag's height.

2.3.1 Nonlinear least-squares — Epoch by epoch

The epoch by epoch approach involves usage of the traditional algorithm used for solving of the non linear least-squares problem. The application of Levenberg-Marquardt (LM) algorithm is described here. The nonlinear problem can be represented in the following form:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \min_{\boldsymbol{x}} \sum_{m=1}^{M} [g_m(\boldsymbol{x})]^2 = \min_{\boldsymbol{x}} \sum_{m=1}^{M} [y_m - \hat{y}_m(\boldsymbol{x})]^2 \,. \tag{2.36}$$

The function f is referred to as the *cost* function, $\boldsymbol{x}$ is the vector of unknown parameters, g_m is a measurement residual, y_m is a measurement and $\hat{y}_m$ is an estimated value of the measurement (output of a model).

The LM solves the problem above in an iterative manner starting from an initial guess $\boldsymbol{x}_0$ until the estimate converges. The update equation for the LM method has the following form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k + \lambda_k \mathbf{I})^{-1}\mathbf{G}_k^{\mathrm{T}} \cdot \boldsymbol{g}(\boldsymbol{x}_k) \,, \tag{2.37}$$

where $\mathbf{I}$ is the identity matrix, λ_k is a positive parameter of the method and $\mathbf{G}$ is a Jacobi matrix of the residual vector $\boldsymbol{g}$

$$\mathbf{G}_k = \left.\frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k} \,. \tag{2.38}$$

The LM method solves the NLSQ problem by combining the advantages of the gradient and Gauss-Newton methods [24, 25]. The method behavior is influenced by the value of the parameter λ_k, which we will explain later. Firstly, we will present the equations for the gradient and Gauss-Newton methods and their properties.

The gradient method finds the function (local) minimum by updating the estimate $\boldsymbol{x}_k$ in the direction of the steepest descent, which is the direction opposite to the gradient of the cost function ∇f. For the estimate update a gradient of the cost function is used, thus the gradient method is a *first order* method

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) \,, \tag{2.39}$$

where α_k is a parameter called step size.

If the problem and cost function are formulated in the sense of NLSQ (2.36), then the gradient has the following form.

$$\nabla f(\boldsymbol{x}_k) = \mathbf{G}_k^{\mathrm{T}} \cdot \boldsymbol{g}(\boldsymbol{x}_k) \tag{2.40}$$

When using the gradient method we may change the step size α_k between iterations, but it is usually held constant for all iterations. Main attribute of the gradient method is that it converges steadily towards a function minimum, but does it rather slowly. With fixed step size, the method may oscillate around the minimum before finally converging to it.

The Gauss-Newton method is a method used specifically to solve NLSQ problems. Its iteration update takes the form of:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k)^{-1}\mathbf{G}_k^{\mathrm{T}} \cdot \boldsymbol{g}(\boldsymbol{x}_k) \,. \tag{2.41}$$

The Gauss-Newton is derived using the second order Taylor series expansion (thus it is a second order method). The inverted expression in the brackets is an approximation of the cost function Hessian $\nabla^2 f$ (the second-derivative matrix) obtained by neglecting the higher order terms. In the equation below we provide the Hessian equation of a cost function having the NLSQ form (Equation (2.36))

$$\nabla^2 f(\boldsymbol{x}) = 2\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k + 2\sum_{m=1}^{M} g_m(\boldsymbol{x}_k)\nabla^2 g_m(\boldsymbol{x}_k)\,. \tag{2.42}$$

The Hessian describes the curvature of the cost function and with this additional information the method dynamically scales and bends the search direction to find a minimum more efficiently.

Gauss-Newton method is faster in achieving a minimum than the gradient method. However, it relies on a good initial guess (close to a minimum) otherwise the estimates tend to diverge.

As already mentioned, the Levenberg-Marquardt method is a combination of the methods described above. Its behavior is changed by manipulating the λ_k parameter. For higher values of λ_k is the inverted expression in the Levenberg-Marquardt update (2.37) approximately equal to $\lambda_k\mathbf{I}$ so the method behaves like the gradient method with a step size of λ_k^{-1}. For lower values of λ_k the identity matrix is negligible in comparison to the other term and the method behaves more like a Gauss-Newton method.

According to [24, 25] the λ_k parameter is changed using the following rules:

- If the cost increases, then the estimate is not accepted as it is further away from the function minimum. The λ_k parameter is increased and the safer gradient behavior of the method is favored.
- If the cost decreases, then the estimate is accepted, since the estimate is converging to a minimum. The λ_k is decreased and the faster Gauss-Newton is favored.

To use the LM method for position estimation we need to formulate the problem in the NLSQ sense, which we have already done with the Equations (2.34) and (2.35), and derive its first derivative. In the calculations we also need to know the positions of the anchors, marked as vectors $\boldsymbol{r}_i$.

$$\min_{\boldsymbol{r}} f(\boldsymbol{r}) = \min_{\boldsymbol{r}} \boldsymbol{g}^{\mathrm{T}}\boldsymbol{g} \tag{2.43}$$

$$g_m(\boldsymbol{r}) = (h_m - (\|\boldsymbol{r}_i - \boldsymbol{r}\| - \|\boldsymbol{r}_j - \boldsymbol{r}\|)) \tag{2.44}$$

The first derivative $\mathbf{G}_m$ (here the subscript denotes the row number rather than the iteration) of the vector element g_m is

$$\mathbf{G}_m = \frac{\partial g_m(\boldsymbol{r})}{\partial \boldsymbol{r}} = \left[(\mathbf{1}_{jr} - \mathbf{1}_{ir})^{\mathrm{T}}\right]\,, \tag{2.45}$$

where the $\mathbf{1}_{ir}$ is a unit vector pointing from the current estimate $\boldsymbol{r}$ to the anchor i.

In a previous section about the anchor synchronization we have stated that with the Kalman filter we are able to obtain the variance of the blink time stamp and with it the quality of the measurement. In the LM equation (2.37) we do not take any measurement variance into account, and so every measurement is treated with equal weight. This may degrade the performance of the estimation if a measurement from a badly synchronized anchor enters the calculations.

It is why the weighting is introduced to the LM algorithm. The weighting acts as a regularization of the inverse term and increases the stability of calculations. More precisely it is the generalized Tikhonov regularization as it follows the same pattern. The measurement weighting is done on the residuals $\boldsymbol{g}$ and the approximated Hessian [26]:

$$\mathbf{G}^{\mathrm{T}}\mathbf{G} \rightarrow \mathbf{G}^{\mathrm{T}}\mathbf{W}\mathbf{G}, \quad \mathbf{G}^{\mathrm{T}}\boldsymbol{g} \rightarrow \mathbf{G}^{\mathrm{T}}\mathbf{W}\boldsymbol{g}\,, \tag{2.46}$$

where weighting matrix $\mathbf{W}$ is a square matrix that is the inverse of the measurement covariance matrix. The equation for the LM update with weighting is following

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\mathbf{G}_k^{\mathrm{T}}\mathbf{W}\mathbf{G}_k + \lambda_k \mathbf{I})^{-1}\mathbf{G}_k^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{x}_k)\,. \tag{2.47}$$

The individual weights are calculated from the blink variance. If the TDoA measurements $\boldsymbol{h}$ were formed using combination matrix $\mathbf{D}$ as

$$\boldsymbol{h} = \mathbf{D}\boldsymbol{t}_{\mathrm{Rx}}c\,, \tag{2.48}$$

then the weights (measurement variances) are calculated in a similar fashion from the vector of time stamp variances $\boldsymbol{\sigma}^2_{\mathrm{Rx}}$, where the individual blink variances are calculated according to the Equation (2.22)

$$\mathbf{W} = (\mathbf{D}\,\mathrm{diag}(\boldsymbol{\sigma}^2_{\mathrm{Rx}})\mathbf{D}^{\mathrm{T}}c^2)^{-1}\,, \tag{2.49}$$

where the operator diag$(\cdot)$ creates a diagonal matrix from the given input vector.

In our implementation we have used the version of the LM update that replaces the identity matrix with a matrix, that has only the diagonal of the matrix $\mathbf{G}_k^{\mathrm{T}}\mathbf{W}\mathbf{G}_k$ [27]

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\mathbf{G}_k^{\mathrm{T}}\mathbf{W}\mathbf{G}_k + \lambda_k \mathrm{Diag}\,(\mathbf{G}_k^{\mathrm{T}}\mathbf{W}\mathbf{G}_k))^{-1}\mathbf{G}_k^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{x}_k)\,, \tag{2.50}$$

where the Diag$(\cdot)$ operator creates a diagonal matrix, nullifying the off-diagonal elements.

In cases where the estimates are far from the minimum and parameter λ_k is large, the influence of the $(\mathbf{G}_k^{\mathrm{T}}\mathbf{W}\mathbf{G}_k)$ becomes negligible to that of $\lambda_k\mathbf{I}$ and the method behaves as gradient-descent method. Replacing the identity matrix with the diagonal of $(\mathbf{G}_k^{\mathrm{T}}\mathbf{W}\mathbf{G}_k)$ helps the estimates to get back to the minimum faster and in the more precise direction, by taking into account the scales of the elements on the main diagonal [27].

2.3.2 Per tag Kalman filter — Continual estimation

The LM method presented in the previous section estimates the tag positions using only the data from the current epoch, making each estimate an isolated one. While this makes the LM method universally usable for every tag, it simultaneously cuts outs any information that may be gained from the connections that inter-epoch estimate possesses.

An approach that uses the inter-epoch information may have more stable (and even accurate) results, for example by taking the tag movement dynamics into consideration. This can be achieved by employing the *Extended Kalman Filter* (EKF) to solve the positioning equations. The EKF also allows us to accept more measurements from different sources into the process. Consequently, we are able to fuse the data and filter the estimated positions during the position estimation process.

Since the positioning problem is nonlinear, the classic linear Kalman filter cannot be applied. The Extended Kalman Filter (EKF) deals with the non-linearity by the means of linearization of the system model around each estimate $\boldsymbol{x}$ [21]. The use of EKF is often challenging for highly nonlinear systems. When the linear approximation is less accurate, the Kalman filter behaves poorly. Therefore, the system must be sampled densely enough.

The non-linearity may be present in the system model and/or the measurement model. Following equations describe both models in the discrete time variant [21]

$$\boldsymbol{x}[k] = \boldsymbol{f}(\boldsymbol{x}[k-1], \boldsymbol{u}[k-1], \boldsymbol{w}[k-1]), \qquad \boldsymbol{w}[k] \sim \mathcal{N}(0, \mathbf{Q}), \tag{2.51}$$
$$\boldsymbol{y}[k] = \boldsymbol{g}(\boldsymbol{x}[k], \boldsymbol{v}[k]), \qquad \boldsymbol{v}[k] \sim \mathcal{N}(0, \mathbf{R}). \tag{2.52}$$

The variables in the above equations have similar meaning as they had in the KF equations in Section 2.1.1. The $\boldsymbol{x}$ denotes the system state, $\boldsymbol{u}$ control input, $\boldsymbol{y}$ system output and $\boldsymbol{w}$ and $\boldsymbol{v}$ the process noise and measurement noise, respectively. The functions $\boldsymbol{f}$ and $\boldsymbol{g}$ capture the nonlinear behavior of the system and measurement models.

The time update step for EKF starts with the computation of the new estimate according to the nonlinear model $\boldsymbol{f}$ with the noise $\boldsymbol{w}$ set to zero.

$$\boldsymbol{x}^{-}[k] = \boldsymbol{f}(\boldsymbol{x}[k-1], \boldsymbol{u}[k-1], \mathbf{0}) \tag{2.53}$$

In order to obtain *a priori* estimate covariance $\mathbf{P}^{-}$, we linearize the above equation at the point of previous *a posteriori* estimate $\boldsymbol{x}[k-1]$.

$$\mathbf{F}[k-1] = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}[k-1],\, \boldsymbol{u}=\boldsymbol{u}[k-1]} \tag{2.54}$$

With the linear matrix $\mathbf{F}$ we are now able to calculate estimate variance using the linear KF equations.

$$\mathbf{P}^{-}[k] = \mathbf{F}[k-1]\mathbf{P}[k-1]\mathbf{F}^{\mathrm{T}}[k-1] + \mathbf{Q} \tag{2.55}$$

Similarly for the measurement update step, the estimated output is obtained by evaluating the nonlinear measurement model $\boldsymbol{g}$ at the *a priori* estimate. Together with the measurement $\boldsymbol{z}[k]$ the innovation is calculated as

$$\boldsymbol{\nu}[k] = \boldsymbol{z}[k] - \boldsymbol{g}(\boldsymbol{x}^{-}[k], \mathbf{0})\,. \tag{2.56}$$

Again, the first partial derivative of the function $\boldsymbol{g}$ (denoted by $\mathbf{G}$) is used for evaluation of the measurement covariance $\mathbf{S}$.

$$\mathbf{G}[k] = \left.\frac{\partial \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^{-}[k]} \tag{2.57}$$
$$\mathbf{S}[k] = \mathbf{G}[k]\mathbf{P}^{-}[k]\mathbf{G}^{\mathrm{T}}[k] + \mathbf{R} \tag{2.58}$$

The *a posteriori* state estimate and covariance are calculated using the following equation [21], which is identical to the one used for the linear KF.

$$\boldsymbol{x}^{+}[k] = \boldsymbol{x}^{-}[k] + \mathbf{K}[k]\boldsymbol{\nu}[k], \quad \mathbf{K}[k] = \mathbf{P}^{-}[k]\mathbf{G}^{\mathrm{T}}[k]\mathbf{S}^{-1}[k] \tag{2.59}$$
$$\mathbf{P}^{+}[k] = (\mathbf{I} - \mathbf{K}[k]\mathbf{G}[k])\mathbf{P}^{-}[k](\mathbf{I} - \mathbf{K}[k]\mathbf{G}[k])^{\mathrm{T}} + \mathbf{K}[k]\mathbf{R}\mathbf{K}^{\mathrm{T}}[k] \tag{2.60}$$

From the presented equations we can see that if both the system and measurement models are linear, then the EKF reduces to linear KF. When the observed system is highly nonlinear, so that the EKF fails to estimate its state correctly, it is better to use the *Unscented Kalman Filter* (UKF). The UKF does not linearize the statistics and tries to preserve the nonlinear character by applying so called *unscented* transformation on the state estimates [28]. We plan to investigate the UKF performance in the following works, however, we do not expect major differences from the EKF performance.

To use the EKF for position estimation we start by determining the state vector $\boldsymbol{x}$ and system model. The model should describe the tag's movement. In the most basic form we can use the model of a stationary object that estimates only object's position $\boldsymbol{r}$, with the system model $\boldsymbol{f}$ being an identity matrix and thus linear.

$$\boldsymbol{x} = \boldsymbol{r}, \quad \mathbf{F} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.61}$$

Between measurements or estimation epochs, the movement of the object will have to be sufficiently slow, because the model assumes a static object. Any movement will then be perceived as a process noise $\boldsymbol{w}$, resulting from the not perfect model and having the covariance $\mathbf{Q}$. Then the matrix $\mathbf{Q}$ has to be carefully tuned to capture the expected movement, in order for the EKF estimates to be able to keep up with the measurements. High blink rate may also help to reduce the model error.

We could add velocity to the model for an improvement of the estimates, for the cases where we are tracking the position of an object that moves in more or less straight lines (such as a trolley, car or a robot). Including the velocity, however, may not be suitable if we are tracking movement of pedestrians. Their movement is more chaotic as they are able reverse the direction of their movement at any time. When this happens, few of the new estimates will still follow the old direction until the filter catches up and corrects the velocity. Therefore, it is more suitable to use the static model for the pedestrian tracking. In the following text we will use this static model, as we will present results from either static or pedestrian movement tracking experiments.

The measurement update step involves a nonlinear equation. As we have mentioned, the TDoA measurements h_m define the set of possible tag positions to be on a hyperbola. Thus the measurement model is the hyperbola equation

$$g_m(\boldsymbol{r}) = \|\boldsymbol{r}_i - \boldsymbol{r}\| - \|\boldsymbol{r}_j - \boldsymbol{r}\| \ , \tag{2.62}$$

where $\boldsymbol{r}$ is the tag's position, $\boldsymbol{r}_i$ is position of anchor i and $g_m(\boldsymbol{r})$ is the predicted output based on the tag and anchors positions. Note that the g_m is the m-th element of the measurement model vector $\boldsymbol{g}$ and that we still use the measurement mapping $\Phi(m) \to (i, j)$ introduced at the beginning of Section 2.3.

The innovation is obtained by comparing the TDoA measurement h_m with the predicted value g_m.

$$\nu_m[k] = h_m[k] - g_m(\boldsymbol{r})[k] \tag{2.63}$$

By linearizing the measurement model $\boldsymbol{g}$ we get the matrix $\mathbf{G}$. For the m-th row of matrix $\mathbf{G}$ it holds

$$\mathbf{G}_m = [(\mathbf{1}_{jr} - \mathbf{1}_{ir})^{\mathrm{T}}] \, , \tag{2.64}$$

where $\mathbf{1}_{ir}$ is a unit vector pointing from the current estimate $\boldsymbol{r}$ towards the anchor i.

It can be observed that the innovation vector $\boldsymbol{\nu}$ is identical to the residual vector in the LM estimation (2.44) and that both methods share the Jacobi matrix $\mathbf{G}$.

The calculation of the measurement covariance matrix $\mathbf{R}$ is almost identical to the calculation of weighting matrix $\mathbf{W}$ (2.49) in the previous section. The difference is that the result for matrix $\mathbf{R}$ is not inverted

$$\mathbf{R} = \mathbf{D} \, \mathrm{diag}(\boldsymbol{\sigma}^2_{\mathrm{Rx}}) \mathbf{D}^{\mathrm{T}} \, c^2 \, . \tag{2.65}$$

The EKF described in this section will serve as a basis for the EKF used for the A2T-TDoA localization in Chapter 3. Some of the results of T2A-TDoA localization are included in Appendix B.

2.3.3 Additional height measurement

For an indoor localization scenario it is common, that the anchors will be placed at the same height, resulting in a flat geometry. Such placement typically results in high vertical DOP and thus the estimate accuracy in the vertical coordinate is rather poor. To cope this problem, it is beneficial to place the anchors in diverse heights, but that is not always possible.

When we are estimating a position of a trolley or a pedestrian, it can be expected that the z-coordinate of the tag will be fairly constant (e.g. tag being placed in a pocket). In such cases it is appropriate to add a soft constraint on the height of the tags to achieve better results in three-dimensional positioning. The soft constraint specifies both the approximate (mean) height p of the tag together with its uncertainty σ_p^2 (variance).

Both the LM estimator and EKF can be easily extended with this constraint, by adding an artificial measurement of the tag's height (or a real one if available). The height measurement in LM is added by appending it to the residual vector $\boldsymbol{g}^{\mathrm{LM}}$

$$\boldsymbol{g}_p^{\mathrm{LM}} = \begin{bmatrix} \boldsymbol{g}^{\mathrm{LM}} \\ p - r_z \end{bmatrix}, \tag{2.66}$$

where r_z is the vertical component of the tag's position estimate and p is the height measurement. The measurement is added to the EKF by expanding the measurement model and the innovation vector

$$\boldsymbol{\nu}_p = \begin{bmatrix} \boldsymbol{\nu} \\ p - r_z \end{bmatrix}. \tag{2.67}$$

The Jacobi matrix $\mathbf{G}$ is shared by both methods

$$\mathbf{G}_p^{\mathrm{LM}} = \begin{bmatrix} & \mathbf{G}^{\mathrm{LM}} & \\ 0 & 0 & -1 \end{bmatrix}, \quad \mathbf{G}_p^{\mathrm{EKF}} = \begin{bmatrix} & \mathbf{G}^{\mathrm{EKF}} & \\ 0 & 0 & -1 \end{bmatrix}. \tag{2.68}$$

Finally the height measurement variance is added

$$\mathbf{W}_p^{\mathrm{LM}} = \left[\begin{array}{ccc|c} & & & 0 \\ & \mathbf{W}^{\mathrm{LM}} & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 1/\sigma_p^2 \end{array}\right], \quad \mathbf{R}_p^{\mathrm{EKF}} = \left[\begin{array}{ccc|c} & & & 0 \\ & \mathbf{R}^{\mathrm{EKF}} & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & \sigma_p^2 \end{array}\right]. \tag{2.69}$$

The performance improvement brought by the soft constraint is presented at the end of Appendix B. More importantly, it is also used for the A2T-TDoA in Chapter 3.

3 Anchor to Tag TDoA

In the previous chapter we have summarized the principles of the *Tag to Anchor* TDoA. This variant estimates the position of the tag from the reception times of a blink message, transmitted by the tag. The measured data is gathered in the so-called *computation* node of the network where the actual estimation takes place. Our focus in this chapter will be on the *Anchor to Tag* TDoA positioning, where the tags can estimate their own position by passively listening to the messages transmitted by the anchors.

The need for a computation node of the T2A-TDoA variant implies that the resulting positions immediately after the estimation are available only to the node (wherever in the network that is). If users need to know their own location, it has to be sent to them from the computation node using a communication channel. Generally, the channel can be the UWB channel, but then the position forwarding would induce a heavy load for the channel and limit the number of tag blinks per second. Therefore, it is not a viable solution for the networks with many users.

As we have written, the T2A-TDoA is capable of handling a hundreds or even a thousand of simultaneously running tags. The limit is determined mostly by the selected UWB channel, channel capacity and the bit rate, with which the blinks are sent. Also the speed of the position estimation is a limiting factor if the real time positioning is required.

One of the advantages of A2T-TDoA is that the position estimate is available to the user immediately after its computation. This is done by allowing the user to measure the data necessary and compute the position on their own. By doing so, it enables the A2T-TDoA to have an unlimited number of simultaneously running tags, which is another advantage of the approach.

The user limit of the T2A-TDoA approach is sufficient for a significant amount of possible use cases. However, the immediate unavailability of the position estimates to the users might be an issue, for example when the system is used for the localization of a robot within a building. Typically for its own function the robot needs to know its position at regular intervals and as soon as possible. Returning the position from the computation node might be problematic if the UWB channel cannot be used for that purpose or if the computation node is not present at all. In that case the A2T-TDoA would be a perfect solution for the robot positioning.

Another example is using the UWB network as an extension to the GNSS localization, providing the positions for both the outside environment and for indoors, where the network is operating. The UWB estimates should be combined with the GNSS estimates seamlessly without the need for the user's intervention and ideally without any additional data load to the UWB network. Thus, the A2T-TDoA seems to be better suited for the problem.

Inspired by the GNSS, the A2T-TDoA variant reverses the messaging scheme between tags and anchors. The anchors now periodically send *beacon* messages that are received by the tags. Similarly to the GNSS the A2T-TDoA can support any number of user tags since the estimation is done at their end and the UWB channel is utilized by the anchors only.

Every tag estimates its own position once a sufficient number of beacon messages is received. Because the tags are listening most of the time of their operation, their power consumption rises significantly as the message reception is more power-demanding than transmission. The consumption is increased even further due to the position estimation. However, given the advantages that A2T-TDoA offers, the increased power consumption is an acceptable price.

For the position estimation we cannot use the algorithms described in Chapter 2 as the error caused by the tag's clock bias (more specifically its bias drift) is no longer negligible.

In GNSS, the positioning signals sent from the satellites use code multiplex [18], which allows the users to receive them simultaneously. Then the data gathered about and from the received messages can directly be used for the position estimation.

The simultaneous full featured message reception, however, is not achievable by the UWB devices, only a single message can be received or transmitted at a time [2, 9]. Therefore the beacon messages have to be transmitted with a delay between each other so that they would not interfere with one another and the tags will have enough time to process them. Yet even when using the shortest delays possible, the bias dynamics of the tag is still capable of rendering the raw measurements useless (as shown in Section 1.2.2).

Nevertheless, we were able to design an EKF that estimates the bias and eliminates its influence on the data and estimates the tag's position as well. The only available methods are able to estimate the position by ignoring the bias problem to some extent, but with much poorer positioning results when compared to our solution.

In the following sections we will discuss the tag clock drift problem in detail. We will describe how the bias influences the measurements and to what extent. Next we will propose a method how to cope with the bias using an *Extended Kalman Filter* and how to estimate the position. At the end of the chapter we will present and discuss the results from the carried out experiments and evaluate the performance of the proposed solution.

3.1 Effect of bias drift on A2T measurements

We have already tackled the topic of the drift effect in the introductory Section 1.2.2 while discussing the TDoA principle. Within a single epoch of the A2T variant, every anchor broadcasts a beacon message at a specified transmit time (this time is included in the data section of the message). The messages are then received by the tag, which also stores the receive time and transmit time of each of the received message. These beacon messages cannot be sent all at the same time, because the UWB devices can receive only one message at the time. Therefore the anchors have to transmit the messages with a delay to one another.

As we consider the TDoA localization principle, we also assume that the time bases of the anchors are synchronized and belong to the same master time domain. Also, the tags have free running clocks and therefore have an arbitrary bias to the master domain.

As in the previous sections, we assume the relation between the transmit and receives times (ToA) to be

$$t^T_{\mathrm{Rx},i} = t^A_{\mathrm{Tx},i} + \tau_i + b_i\,, \tag{3.1}$$

where the super script denotes the time domain of the measurement, (T) for tag and (A) for anchor, τ_i is the propagation time between the two devices and the b_i bias of the two time domains. In general, the bias b_i is not stationary and changes in time.

In the calculations of the TDoA measurements, the tag has to compensate for the transmission spacing T_{ij} by subtracting it from the measurement.

$$\tilde{h}_m = (t^T_{\mathrm{Rx},i} - t^T_{\mathrm{Rx},j}) - T_{ij} \tag{3.2}$$

Since the anchors are synchronized, the transmission spacing can be derived from the message transmission times.

$$T_{ij} = t^A_{\mathrm{Tx},i} - t^A_{\mathrm{Tx},j} \tag{3.3}$$

Then we can form the TDoA measurement by rearranging the Equation (3.1)

$$t^T_{\mathrm{Rx},i} - t^A_{\mathrm{Tx},i} = \tau_i + b_i\,, \tag{3.4}$$

and taking the difference with data from two anchors.

$$\tilde{h}_m = (t^T_{\mathrm{Rx},i} - t^T_{\mathrm{Rx},j}) - (t^A_{\mathrm{Tx},i} - t^A_{\mathrm{Tx},j}) = (\tau_i - \tau_j) + (b_i - b_j) \tag{3.5}$$

From the (3.5) we can see how the tag time base bias contributes to the measurements. In contrast to the T2A variant, neither the transmit times nor the biases cancel out, since they are not equal and measured at different instances. The anchors are synchronized and therefore we can model how the tag clock dynamics differs from the anchor clock dynamics. If we use the linear model (1.3), we can express the bias difference using the tag bias drift $\dot{b}$ w.r.t. the anchor clock drift.

$$b_i - b_j = \dot{b} \cdot (t^T_{\mathrm{Rx},i} - t^T_{\mathrm{Rx},j}) \tag{3.6}$$

It is important to clarify that it does not matter whether we use, for the (3.6), the difference of receive times in tag's domain or transmit times in master domain. The two time differences will differ in the order of nanoseconds, due to the propagation times in reception time difference and bias drift. Therefore the transmission spacing (order of milliseconds) present in these differences will be more dominant. After multiplying the time difference with the bias drift $\dot{b}$, the two differences (difference of reception times and difference of transmission times) will become indistinguishable and both will express the same transmission spacing of the beacon messages. This has been also proven in [4].

For the position estimation we would rather work with distances and for that, we multiply the TDoA measurement $\tilde{h}_m$ by the signal propagation speed c to convert the time differences into the distance differences.

$$\begin{aligned} h_m &= c \cdot [(t^T_{\mathrm{Rx},i} - t^T_{\mathrm{Rx},j}) - (t^A_{\mathrm{Tx},i} - t^A_{\mathrm{Tx},j})] \\ &= c \cdot (\tau_i - \tau_j) + c \cdot (b_i - b_j) \\ &= \underbrace{(\|\boldsymbol{r} - \boldsymbol{r}_i\| - \|\boldsymbol{r} - \boldsymbol{r}_j\|)}_{\text{TDoA}} + \underbrace{c \cdot \dot{b} \cdot (t^T_{\mathrm{Rx},i} - t^T_{\mathrm{Rx},j})}_{\text{Tag clock drift error}} \end{aligned} \tag{3.7}$$

The bias drift comes into the TDoA measurement as an error source. According to the IEEE 802.15.4 [2] the crystal oscillators used for the UWB devices have their bias drifts in the range ±20 ppm. Using the highest bit rate offered by DW1000, which is 6.8 Mbps [9], each beacon message should be transmitted within a millisecond. Then we can expect the transmission spacing (or the delay between message receptions) to be in units of milliseconds.

In Table 3.1 we demonstrate how the clock drift and choice of transmission spacing influences the measurement error. As show in the table, even with very precise oscillators and short transmission spacing the error easily reaches order of meters. If we take the worst case for the oscillator drift and spacing of 10 ms, the error has the magnitude of hundreds of meters, which makes the measurements useless.

The transmission spacing should be as low as possible for two reasons. First is to reduce the measurement error and second to ensure that the linear approximation of the bias dynamics will be as close as possible. From the other perspective, the transmission spacing has to be long enough for the tags to be able to process the message and prepare for reception of another message.

Either way we have to compensate for the bias effect to be able to estimate positions using the A2T-TDoA. Since we are not really interested in the absolute bias between tag and anchor time scale[11] we can use an approach with the Kalman filter as we did in Section 2.1.1 and estimate only its dynamics, the bias drift $\dot{b}$ and its rate of change $\ddot{b}$. With the estimated drift we could fix the measurements and use EKF for position estimation, derived in Section 2.3.2.

Such disconnected approach (separation of bias and position estimation) could face some issues with the stability of the estimates, since the two, bias and position, are clearly firmly connected as can be seen in (3.5) and (3.7). A more stable solution is to expand the *Extended Kalman filter* from Section 2.3.2 by adding the bias drift $\dot{b}$ and drift rate $\ddot{b}$ to its state vector and jointly estimate position and bias. The derivation of such EKF will be further in this chapter.

3.1.1 Obtaining A2T measurements

To determine the A2T measurements, the tag listens to the beacon messages, that are periodically sent by the anchors. The beacon messages contain the time of their transmission from the source anchor and each message has upon reception assigned also a receive time.

The structure of the beacon messages is nearly identical to the synchronization or relay messages. From the section about the Chained synchronization 2.1.2 we know, that the relay message additionally contains the variance of the transmission time, which is used for the synchronization of the slave anchors but it can be also used for the tag's position estimation. The variance will prove to be rather useful.

Because the relay messages already contain all the data necessary for the A2T measurements (transmit time and variance) it is beneficial to use them also for the position estimation. If all the anchors are configured as relays, with an appropriate transmission spacing, the tags can then listen for the relay/beacon messages and position themselves from the captured data. Using this scheme, the tags "parasite" on the synchronization network and passively localize themselves, without anyone's knowledge. This is illustrated by Figure 3.1, where there is one master anchor M, two relays R_1 and R_2, one slave anchor and one tag. The synchronization

Table 3.1: Effect of clock drift and transmission spacing on the measurement error

		Transmission spacing		
		1 ms	10 ms	1 s
Drift	1 ppm	0.3 m	3 m	300 m
	3 ppm	0.9 m	9 m	900 m
	20 ppm	6 m	60 m	6 km
	40 ppm	12 m	120 m	12 km

[11]We are not able to measure the bias, because we would have to know the distance between the tag and an anchor (see (3.1) and (2.15)) which we do not know prior to the position estimation.

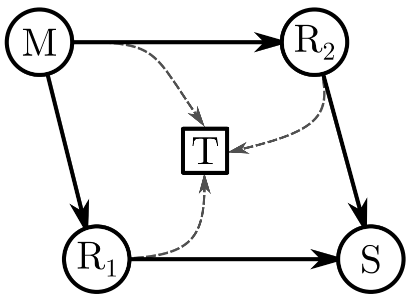


Figure 3.1: Tag parasiting on the synchronization messages

messages are symbolized as the solid lines, whereas the dashed ones are used to represent that the tag can receive those messages as well.

In this section we have used the terms relay and beacon messages to name basically identical message. To avoid the confusion we will use the term *relay* message in the context of the Chained synchronization and *beacon* message in the context of A2T-TDoA localization.

3.2 Estimation of position and drift

In A2T-TDoA each measurement contains an error induced by the free running tag clock. For a successful position estimation this error source has to be compensated for.

In the context of TDoA-UWB positioning this topic is not yet very well explored even though it might be intriguing and useful for self-navigating robots or small vehicles. The reason is probably because of the troubles connected to the drift and also because T2A-TDoA variant is easier to implement and its properties are usually sufficient (total user count and position availability).

From the available sources three interesting algorithms are presented, each one of them unique in its approach. It is interesting that in each of the papers, the authors used the DW1000 UWB chip for the UWB communication and positioning, the same model used by us.

First of the articles [29] by Corbalán et. al. presents a system called Chorus, which is able to simultaneously receive multiple messages with the DW1000 hardware and mitigate the problem with the bias. They are able to get time stamps of several simultaneously arriving messages even though it is not defined by the standard IEEE 802.15.4 [2]. They do so by examining the *Channel Impulse Response* (response of a matched filter) right after the reception of the first message. However the message reception is not full-fledged as they are able only to detect whether a message has arrived and when. By the proposed procedure it is not possible to retrieve any other information carried by the message. Also it is not possible to identify the source of a message just by the channel response alone. Therefore the message scheduling has to be passed to the tag beforehand.

Second system [30] called SnapLoc and developed by Großwindhager et. al. ignores the problem with the bias drift of the user tag by using extremely short transmission spacing (128 ns) between the localization messages. Authors also use a dedicated anchor for synchronization, however this anchor does not participate in the localization (does not send the localization messages). Similarly to the Chorus [29], also SnapLoc [30] struggles with delivering data in the localization messages. The information needed for localization and identification of the localization message has to be passed in advance.

Finally the third approach, which is conceptually closest to ours, has been developed by Ledergerber and his team for a mobile robot self-localization [31]. In their paper each localization epoch is composed of two steps. Firstly the synchronization message for the anchors is sent from the reference anchor. In the second step the reference anchor sends a localization message, which is then forwarded by the other anchors with a given delay (1 ms) between each transmission. Each of the sent messages (synchronization and localization) is also received by the tags. Using data from the messages sent by the reference anchor, tag firstly estimates its bias drift and then proceeds to estimate its position using the EKF.

We can see that the approaches are quite diverse, each trying to solve the problem differently. However, all of the approaches are struggling to a certain extent with the scalability of the localization area. Authors do not discuss how the anchors are synchronized beyond the room or hall where the reference anchor is located. The aforementioned approaches also set a hard limit on the maximal number of anchors used for the localization. For example SnapLoc supports eight anchors at maximum [30] for a given transmission spacing. As the author states, if more anchors are needed, the transmission spacing has to be shortened, but the precision worsens as the measurements are more susceptible to the multipath [30].

Our solution, which we will present in this section, is a novel method how to localize with A2T-TDoA variant with results and precision comparable to that of the T2A and preserving the scalability that the original system has. Also the number of anchors, which can provide localization data, has bound expected to be in tens. The anchor limit is set by the length of synchronization period and transmission delay between beacon messages.

We will also discuss the differences between our solution and the solution provided by [31] at the end of this section.

3.2.1 Extended Kalman Filter

For the position estimation from the bias-affected A2T measurements we can use the EKF, which we have implemented in Section 2.3.2, and modify it to estimate also the bias dynamics. We begin by adding the bias drift $\dot{b}$ and its rate of change $\ddot{b}$ to the filter state vector.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}^{\mathrm{T}} & \dot{b} & \ddot{b} \end{bmatrix}^{\mathrm{T}} \tag{3.8}$$

Next, we define the system model. For the bias drift we will use the discrete linear model introduced in (1.3) and because we do not need the bias[12] b we can omit it and use a reduced version of the model

$$\dot{b}[k] = \dot{b}[k-1] + \ddot{b}[k-1]\,T_e\,, \tag{3.9}$$

where T_e is time elapsed between discrete times $k-1$ and k.

With the bias drift model and the movement model at hand we can now form the state transition matrix $\mathbf{F}$

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & \mathbf{0}_{3\times 2} \\ \mathbf{0}_{2\times 3} & \begin{matrix} 1 & T_k \\ 0 & 1 \end{matrix} \end{bmatrix}, \tag{3.10}$$

where T_k is a time delay between epochs $k-1$ and k.

[12] In fact, we are unable to measure it as it is an unobservable state, due to the differentiation in TDoA.

The measurement model is derived from (3.7). Here we observe the state vector through the measured times of arrival and transmission

$$z_m[k] = c \cdot [(t^T_{\mathrm{Rx},i}[k] - t^T_{\mathrm{Rx},j}[k]) - (t^A_{\mathrm{Tx},i}[k] - t^A_{\mathrm{Tx},j}[k])], \tag{3.11}$$

which can be written in the vector form with the use of the combination matrix $\mathbf{D}$

$$\boldsymbol{z}[k] = \mathbf{D} \cdot (\boldsymbol{t}^T_{\mathrm{Rx}}[k] - \boldsymbol{t}^A_{\mathrm{Tx}}[k]) \cdot c\,. \tag{3.12}$$

The measurement model is an augmented version of the measurement model (2.62), we have derived for the T2A EKF. The model contains an extra term related to the bias drift effect.

$$y_m[k] = (\|\boldsymbol{r}[k] - \boldsymbol{r}_i\| - \|\boldsymbol{r}[k] - \boldsymbol{r}_j\|) + c \cdot \dot{b}[k] \cdot (t^T_{\mathrm{Rx},i}[k] - t^T_{\mathrm{Rx},j}[k]) \tag{3.13}$$

After the linearization of the measurement model we get the matrix $\mathbf{G}$. For each row $\mathbf{G}_m$ of the matrix it holds that

$$\mathbf{G}_m = \begin{bmatrix} (\mathbf{1}_{i,r} - \mathbf{1}_{j,r})^{\mathrm{T}} & t^T_{\mathrm{Rx},i}[k] - t^T_{\mathrm{Rx},j}[k] & 0 \end{bmatrix}. \tag{3.14}$$

The measurement covariance is determined in a similar way as it was for the T2A EKF in Section 2.3.2. In the case of A2T the measurements are formed using both the receive times and the transmit times, so the variance of the transmission times $\boldsymbol{\sigma}^2_{\mathrm{Tx}}$ has to be added to the measurement covariance. The transmission covariance is contained within data part of the beacon messages so that it is available for the tag. The resulting covariance $\mathbf{R}$ is

$$\mathbf{R} = \mathbf{D} \cdot \mathrm{diag}(\boldsymbol{\sigma}^2_{\mathrm{Rx}} + \boldsymbol{\sigma}^2_{\mathrm{Tx}}) \cdot \mathbf{D}^{\mathrm{T}} \cdot c^2\,, \tag{3.15}$$

where $\boldsymbol{\sigma}^2_{\mathrm{Rx}}$ is again the variance of a time measurement, and operator $\mathrm{diag}(\cdot)$ creates a diagonal matrix from the input vector.

With the described EKF the tag is able to estimate its own position once it collects a sufficient amount of A2T measurements (using at least four beacon messages for the three-dimensional estimate).

However, the performance of the estimate might not be the best one achievable yet, because we are trying to estimate both the position and the bias drift dynamics using a single type of measurements. If we use only the sufficient number of A2T measurements (three for in a 3D half-space), then the resulting set of equations (measurements) would be undetermined because of the unknown bias drift $\dot{b}$. The estimation would perceive the drift as a free parameter or variable. To cope with this we may use more measurements that it is sufficient for the position estimation. But in that case there is still an issue that we are observing the drift dynamics only on a short time intervals, i.e., the transmission delays between beacon transmissions.

To observe the bias time development more precisely it is better to observe it on wider intervals. To do this we add an auxiliary measurement that helps us to measure the bias development on wider scale by using the data obtained in successive epochs.

3.2.2 Auxiliary drift measurement update

In order to get a better observation of the bias dynamics we have to extend the interval on which the bias measurements are done. It can be achieved by providing an additional measurements to the EKF. As the tag is receiving the beacon messages periodically we can use the data received in consecutive epochs for the estimation.

With each received beacon message we are given the time of message transmission and reception. We remind that for the two times the following holds

$$t^T_{\mathrm{Rx},i}[k] - t^A_{\mathrm{Tx},i}[k] = \tau_i[k] + b_i[k]\,, \tag{3.16}$$

where the index k marks the epoch when the measurement was taken.

In the (3.16) the bias b_i, between the tag and anchor time bases, is present and since we are unable to observe or estimate the bias we cannot use this equation directly for the bias measurement. But we are able to measure the change of the bias by taking the difference of reception and transmission times between subsequent epochs

$$(t_{\mathrm{Rx,i}}[k] - t_{\mathrm{Rx,j}}[k-1]) - (t_{\mathrm{Tx,i}}[k] - t_{\mathrm{Tx,j}}[k-1]) = \tau_i[k] - \tau_j[k-1] + b_i[k] - b_j[k-1]\,. \tag{3.17}$$

By taking the difference of data from two consecutive epochs we get an observation of how much the bias and the time of flight have changed from the last epoch. Of course, nothing forbids us to use data originating from different anchors, but as the difference of the propagation times $\Delta\tau$ (calculated from tag's position estimate and anchors positions) appears in the Equation (3.17) it would again make the measurements useless as we do not know the propagation times prior to the estimation. We would rather omit the estimated propagation times from the measurements entirely as well, because correctly observing the bias dynamics is a crucial part of the A2T positioning and we want to get rid of any potential error sources. If we would include the estimated propagation times with the intention of getting better measurements we would more likely disturb the bias measurements. It is because the estimates would not be precise enough (especially during the first epochs), making the positioning needlessly much more difficult for the estimation process. Therefore, we create such bias measurements using only the data from messages sent by the same anchor (such that $i = j$).

By measuring the bias change using data from a single anchor only, we can express the differences as

$$\tau_i[k] - \tau_i[k-1] + b_i[k] - b_i[k-1] = \Delta\tau + \Delta b\,, \tag{3.18}$$

where we express the change of bias Δb with states $\dot{b}$ and $\ddot{b}$

$$\Delta b = T_k\dot{b} + \frac{1}{2}T_k^2\,\ddot{b}\,, \quad T_k = t^T_{\mathrm{Rx},i}[k] - t^T_{\mathrm{Rx},i}[k-1]\,. \tag{3.19}$$

The Equation (3.18) in this form could be used as a bias measurement model for the EKF, but it still contains the unknown term $\Delta\tau$, which we cannot eliminate with the estimated values for the reasons discussed above. Here, we are forced to make an assumption that the change in τ between epochs is negligible or within the bounds set by the measurement covariance.

We suspect that this assumption might be correct, because the propagation time difference $\Delta\tau$ is associated only with one anchor. Let us assume the following example where we are estimating the position of a slowly moving object, for example a pedestrian or a fork lifter, with speed up to $10\,\mathrm{km\,h^{-1}}$ and with a delay between epochs $T_k = 100\,\mathrm{ms}$. Then we can expect the propagation time difference from an anchor to the tag to be in units of nanoseconds. Now if we assume the tag's bias drift to be 40 ppm and stable (after warm up), then with the same epoch delay $T_k = 100\,\mathrm{ms}$ the change of bias should be in units of microseconds.

By comparison we see that for slowly moving objects the change of bias Δb is by three orders of magnitude greater than the change of propagation time $\Delta\tau$ and that the assumption $\Delta b \gg$

$\Delta\tau$ holds. Furthermore, the assumption holds also for objects moving at higher velocities (close to $100\,\mathrm{km\,h^{-1}}$) as the $\Delta\tau$ is does not exceed 20 ns.

Under this assumption we can ignore the $\Delta\tau$ term, which leads us to the bias measurement model $\boldsymbol{g}^b[k]$. The model is linear and for a measurement with an index m

$$\boldsymbol{y}^b_m = \left(t_{\mathrm{Rx,i}}[k] - t_{\mathrm{Rx,i}}[k-1]\right) - \left(t_{\mathrm{Tx,i}}[k] - t_{\mathrm{Tx,i}}[k-1]\right), \tag{3.20}$$

the model has the following form

$$\boldsymbol{g}^b_m[k] = \underbrace{\begin{bmatrix} 0 & 0 & 0 & T_k & \frac{1}{2}T_k^2 \end{bmatrix}}_{\mathbf{G}^b_m} \cdot \boldsymbol{x}^-[k], \tag{3.21}$$

where the $\mathbf{G}^b_m$ is the m-th row of the measurement model matrix $\mathbf{G}^b$, T_k is the delay between reception of the two beacon messages from the same anchor and $\boldsymbol{x}^-[k]$ is the *a priori* state estimate in the epoch k.

3.2.3 Hybrid measurements

By adding the auxiliary bias measurement we have augmented the EKF to have more independent inputs to estimate the bias dynamics, both on a short intervals and the longer intervals.

The EKF is by its definition able to accept both measurement types in a single measurement update and perform data fusion on them. However, this approach would face numerical problems because the position measurements (3.12) are numerically much larger (units to tens of meters) than the bias measurements (ppm). The same applies for the corresponding measurement variances. This results in the combined innovation covariance $\mathbf{S}$ being ill conditioned (due to matrix containing both large and small values on the diagonal) and being unable to accurately compute its inverse in (2.59). To cope with this we have to split the measurement update into two parts, one for bias update and one for position update.

Another argument for splitting the measurement update is that the bias update is independent of the position estimate (we have removed the dependency from the model (3.21)), whereas the position update is dependent on the bias estimate. Therefore the bias measurement update should be done before the position update. Having two measurement inputs gives us also the possibility to perform the bias measurement update even when we do not have enough measurements to estimate the position, but enough of them to estimate bias.

In the next section we provide the results of a series of experimental tests done with the proposed EKF. The tests carried out were focused on the performance of position estimation as well as the contributions of the auxiliary measurements.

3.3 Experimental results

This section contains the results of the A2T-TDoA positioning. We present data from three of the concluded experiments, two tests with tag static and one test with moving tag. All experiments were situated in a classroom, 4 m wide and 10 m long, with six anchors and one tag. The tag was mounted on top of a tripod (height 1.6 m) with its position perfectly known during the static tests. The anchors were mounted in a grid like structure, all at the same height of 2.5 m.

The presented data from static experiments focuses on bias estimation, importance of the auxiliary bias measurements and the performance of the positioning in specified locations within the network. The static experiments have the height soft constraint included.

The dynamic measurement presents the positioning performance of a moving tag and how the height soft constraint improves the A2T-TDoA results, if the anchor geometry is not favorable for height estimation.

The first test was performed with the tag mounted on top of a tripod and placed near the center of the room. The tag remained stationary during the measurement. Its height has been constrained to 1.6 m with variance $(25\,\mathrm{cm})^2$.

First, we will discuss the effect of the clock on the TDoA measurements and its elimination. The TDoA measurements without the bias corrections are plotted in Figure 3.2a. For a static object, the TDoA measurements should be almost constant. However, due to the changing clock drift they change in time as well, as can be seen in Figure 3.2a. Figure 3.2b depicts the estimated bias drift and drift rate. By comparison of the drift evolution and evolution of the raw TDoA measurements, we can conclude that the drift influences the measurements, as they follow the trend set by the drift. As a consequence of the drift effect, the raw TDoA measurements are far larger than anticipated[13] for the room of size 4 m by 10 m. The expected size of measurements is in order of meters (below 10 ns), not tens of meters (over 30 ns) as seen in Figure 3.2a.

The EKF, proposed in this chapter, corrects the TDoA measurements using the estimated bias. Afterwards, the EKF estimates position. The corrected measurements can be seen in Figure 3.2c, which seem to have almost constant mean value with standard deviation approximately 500 ps (15 cm). Figure 3.2d shows the top-down perspective of the classroom with the estimated positions. From this view, it can be observed that the estimated positions are very close to the true position of the tag. This is proven by the mean estimation error μ_ε and its variance σ_ε^2, which is

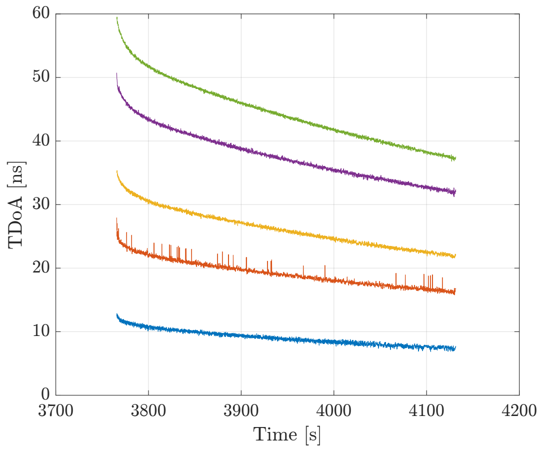
$$\mu_\varepsilon = 20.1\,\mathrm{cm}, \quad \sigma_\varepsilon^2 = (3.9\,\mathrm{cm})^2\,. \tag{3.22}$$

Figures 3.3 and 3.4 show the importance of auxiliary bias drift measurements. By comparing the estimates of the drift in Figure 3.3, we can see that without the auxiliary measurements they differ mostly in the estimation of the drift rate $\ddot{b}$. In such circumstances the EKF fails to observe the change in clock drift correctly, as it has only the short scale measurements of the drift, where it remains fairly constant. This is supported by the fact that the drift rate estimate $\ddot{b}$ in the Figure 3.3a cannot be labeled as the derivative of the clock drift $\dot{b}$, as the drift rate is zero while the drift $\dot{b}$ is clearly changing.

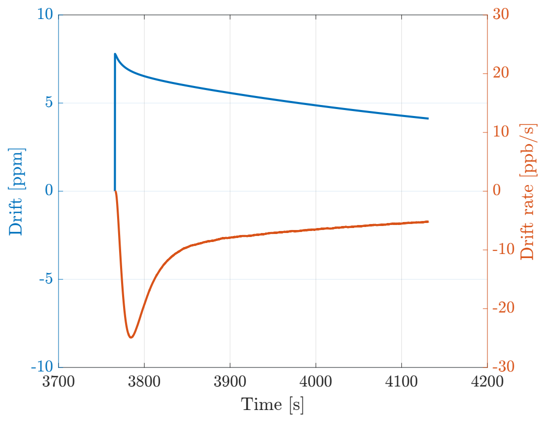
Figure 3.4 compares the estimated x, y and z. The estimated coordinates are presented with their mean value and 95 % confidence intervals, provided by the estimate covariance matrix $\mathbf{P}$ from the EKF. In Figure 3.4a, it can be seen that due to the poor estimation of the drift $\dot{b}$ and drift rate $\ddot{b}$ the coordinates are not constant, as opposed to the correctly estimated coordinates in Figure 3.4b. The error in the position estimation is most notable in the z-coordinate, where its value goes as low as 1 m before returning back up. The drop of the value corresponds to the interval, when the estimated drift rate was zero, while the drift was clearly changing.

In the next experiment we performed several static measurements on a grid in the classroom. During the measurements both the A2T and T2A measurements were collected and

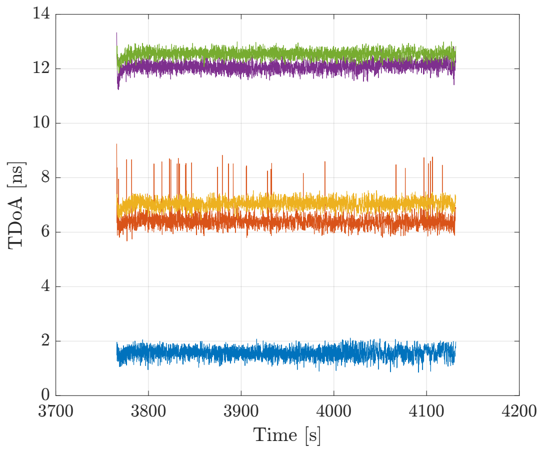
[13]The TDoA between anchors i and j should not be larger than the distance between anchors i and j.



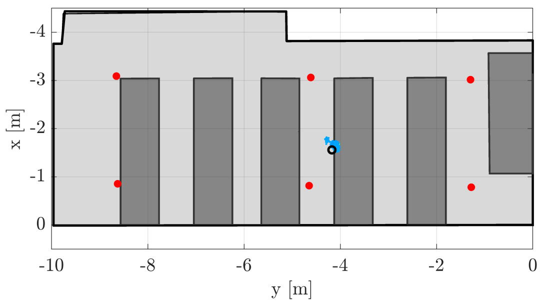
(a) Measured TDoA h_{ij} without drift corrections



(b) Estimated drift and drift rate



(c) Measured TDoA h_{ij} with drift corrections



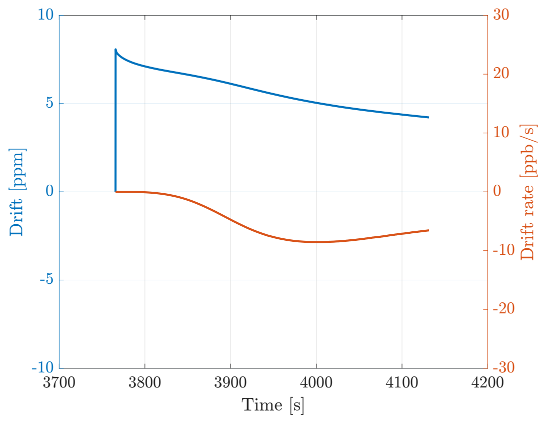
(d) Estimated position

Figure 3.2: Static test 1: Measured data and estimation results

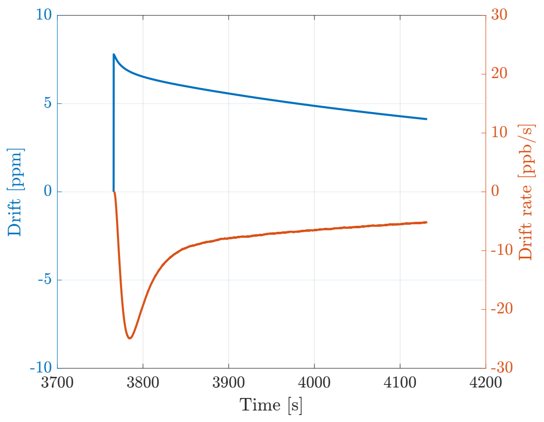
the estimation was done with height soft constraint. This test provides the comparison of the positioning performance in different places of an area and also the comparison of the A2T and T2A positioning.

The estimation results can be seen in Figure 3.5, where the anchors are depicted as black circles, tag's ground truth positions as black crosses and estimated positions as dots, with color indicating position's total standard deviation. Both methods achieved best performance when the tag was inside the anchor constellation. When the tag was placed outside of the constellation, the performance deteriorated as the DOP increased.

The two methods are comparable based on their performance and estimate error. However, there are differences. For example, the estimates in the first row from the left are more stable for A2T then they are for T2A. This may be caused by the increase of measurement error due to the higher SNR of the positioning messages (either beacon or blink), which is caused by unfavorable mutual orientations of the anchors and tag antennas. The relations between measurement error and received power is investigated by [22] and will be discussed in future works.

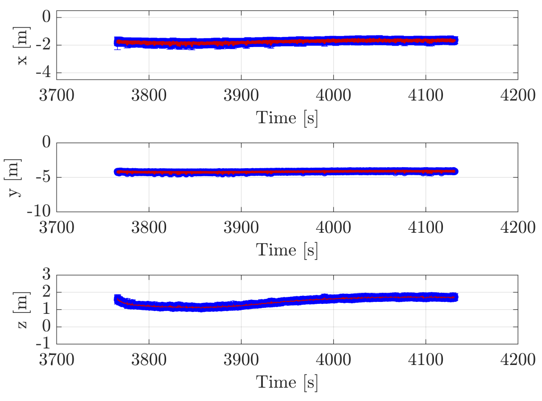


(a) Without auxiliary measurements

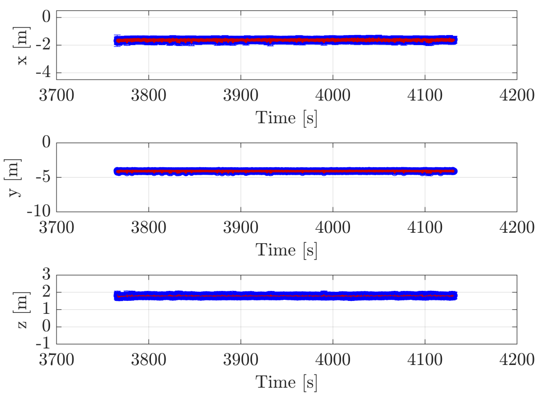


(b) With auxiliary measurements

Figure 3.3: Static test 1: Drift estimate with and without auxiliary measurements



(a) Without auxiliary measurements



(b) With auxiliary measurements

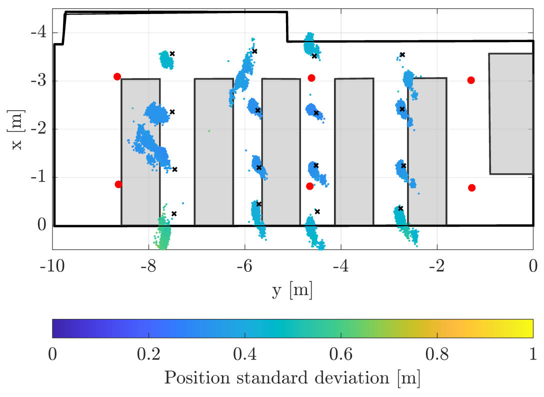
Figure 3.4: Static test 1: Coordinate plot with and without auxiliary measurements (red is estimated value, blue confidence interval)

The results are summarized by Table 3.2, where the total RMS and mean error is stated for both methods.

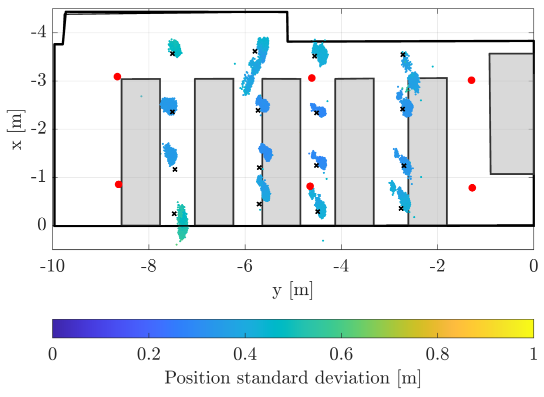
Finally, the last test was focused on the dynamic positioning performance of the A2T-TDoA and how the height soft constraint may improve it. During this test, the tag was attached to a tripod and the movement was realized by moving the tripod. The tripod was moved by hand between the desks in an E-like shape. The results can be seen in Figure 3.6, where the desks are depicted as gray rectangles.

Table 3.2: Static test 2: Estimation error comparison

	RMS [m]		Mean error [m]		
	2D	3D	x	y	z
Anchor to Tag	0.26	0.30	-0.03	0.10	0.08
Tag to Anchor	0.34	0.37	0.10	-0.12	-0.03

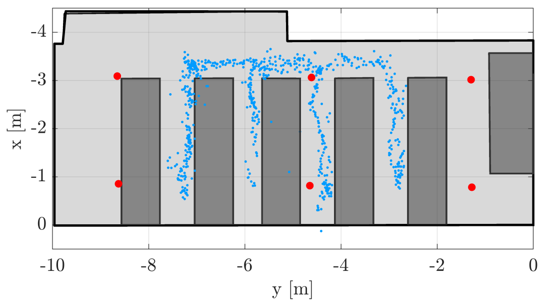


(a) Tag to Anchor

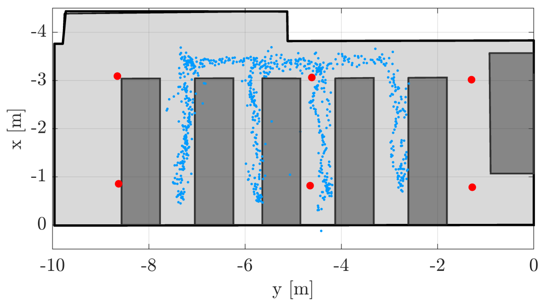


(b) Anchor to Tag

Figure 3.5: Static test 2: Grid, estimated positions



(a) Height unconstrained



(b) Height constrained to 1.6 m

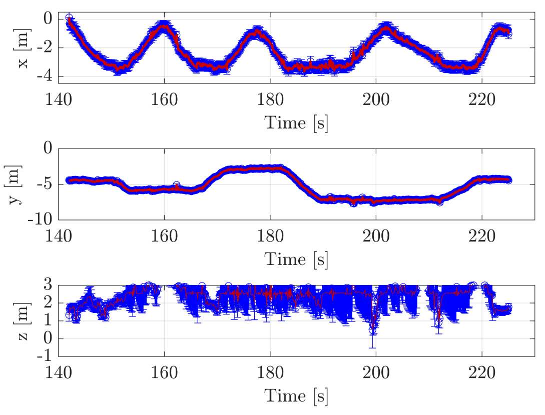
Figure 3.6: Dynamic test: Estimated positions

By looking at the top-down view in Figure 3.6, it can be seen that the unconstrained estimation (Figure 3.6a) has the estimate less dense, in comparison to the constrained estimation (Figure 3.6b). This is because the estimates that were higher than 3 m were cropped from the plot. The higher estimation error in the z-coordinate is a result of flat geometry of the anchors. As can be seen in Figure 3.7a, the mean value of the estimated z-coordinate is mostly higher than 2 m, but with large confidence intervals that span even to the 1.6 m values.

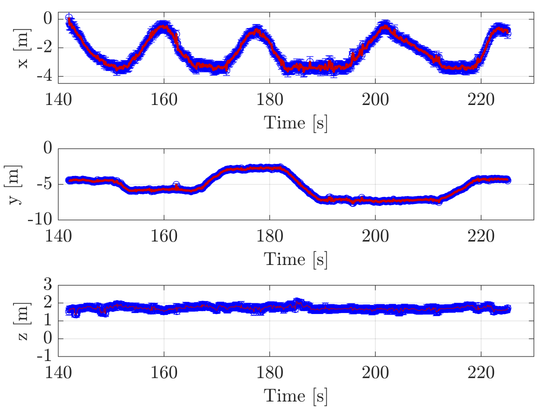
By adding the height soft constraint, we were able to reduce the variance of the z-coordinate estimates, which then improved the estimation in the horizontal plane.

3.4 Perspectives of the A2T positioning

In this chapter we have presented an implementation of *Anchor to Tag* TDoA positioning for UWB networks using the Extended Kalman Filter for the position estimation. In A2T-TDoA variant the stationary and synchronized anchors periodically broadcast beacon messages that are received by the tags. Each tag is able to estimate its own position in real-time using these messages. Also, there is no limitation on how many tags can receive the beacon messages, enabling any tag within the network to position and possibly navigate itself. This of course comes with the cost of increased power consumption of the tags, due to the listening for the messages and additional computations.



(a) Height unconstrained



(b) Height constrained to 1.6 m

Figure 3.7: Dynamic test: Position in time with 95 % confidence intervals

While the A2T-TDoA variant provides appealing advantages (real-time position, unlimited users), it is still a concept that is relatively new to the UWB-TDoA networks. The A2T-TDoA also faces great challenges posed by the free-running nature of the tag's clocks. Thus, the resources concerning the A2T are rare.

Nonetheless, several articles that implement the A2T-TDoA positioning do exist, each choosing a different approach. The solutions presented by the available articles [29–31], have to accept several restrictions to assure proper function of their approach. The most notable restrictions are the positioning scalability and the inability to send data in the positioning messages [29, 30].

The approach proposed in [31], where they estimated position of a quadcopter, is conceptually very similar to ours. The authors also used the EKF for position estimation and they also used an one-way synchronization. Also, the presented algorithm combines UWB measurements with accelerometer and barometer data, improving the estimation of z-coordinate and velocity [31]. However, their A2T localization relies on an extended synchronization scheme, where every epoch an extra synchronization message is sent. These two messages are used for estimation of the tag's clock drift. The second synchronization message is relayed by the anchors to the tag and used for its positioning.

The algorithm proposed by [31] differs from ours in several aspects. First of all, our algorithm requires only one synchronization message to be sent per epoch and anchor (without the extra message for drift estimation). Second, we have improved the drift measurements by measuring it on larger intervals, whereas [31] measure the drift only on 1 ms intervals.

The article [31] provides the estimation results from the quadcopter flight. For evaluation they used so-called mean RMS of the estimation error. Which is not really an appropriate metric as it is the mean size of position error

$$\text{mean RMS} = \frac{1}{N}\sum_{i=0}^{N}\sqrt{[(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2]}\,, \tag{3.23}$$

while the RMS calculates the square-root of mean quadratic position error

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}[(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2]}\,, \tag{3.24}$$

where $\hat{x}_i, \hat{y}_i$ are estimated coordinates, x_i, y_i are the corresponding true values and N is the number of measurements.

Nevertheless, using this metric they achieve mean RMS error of 14 cm in horizontal plane. However, they also fuse the UWB measurements with the data from an accelerometer and barometer, which improves their estimation. Due to the data fusion and that we do not have the ground truth for the dynamic test, the performance is not comparable with our solution. For the sake of completeness, using similar metric on the static grid test with constrained height we achieve mean RMS error of 15 cm in horizontal plane.

The estimation A2T-TDoA results, presented in the previous chapter, conclude that the EKF algorithm proposed in this chapter achieves great accuracy when positioning both stationary and moving objects. It should be noted that the A2T can be used in parallel with T2A. The fact that anchors behave as relays, does not collide with their ability to receive blink messages and send them to the computation node for estimation. This behavior can be exploited in situations when the position of certain objects has to be tracked, while certain users require their real-time position.

In the future works we want to investigate the use of Unscented Kalman Filter (UKF) instead of EKF and, given that tags are able to estimate clock drift, also the possibility of utilizing ToA measurements for position estimation.

4 Automated determination of the anchor network

In the previous chapters we have described the algorithms for position estimation in the UWB-TDoA networks, using either the T2A or A2T approach. The devices which partake in the localization are the anchors and tags. The anchors are nodes with fixed and known position, while the tags are mobile user nodes with position determined during the position estimation.

The anchor properties suggest that the network is established well before the estimation and that we already have the knowledge of the anchor positions. Nonetheless, such knowledge is usually unknown to us, because in most cases we are the ones who create the network by placing the anchors and thus we have to determine the anchor positions ourselves. The determination of the positions commonly involves specialized equipment for distance measurement and positioning. Such approach can be time consuming, especially for large networks inside a diverse environments (such as building interiors), or not feasible enough for smaller, one-time use networks.

The usual tools for determination of anchor positions involve some kind of distance measurement. However, the anchors themselves already have the means to measure distance between each other. Using the TWR protocol (see Section 1.2.1).

In this chapter we will describe the algorithm proposed by [32] for anchor self-positioning, and suggest few improvements of that algorithm. Main advantage of these algorithms is that the calculations are distributed and are done by the anchors. Consequently, there is no need for a computation node dedicated for the calculations.

The content of this chapter will be more practical in a sense that it will focus on the problems in localization networks that are commonly not tackled. In the first section we will discuss the common methods of the anchor position determination along with the necessary basics of graph theory. The following sections will present several algorithms that can solve the problem. First will be a non-distributed Levenberg-Marquardt, which we will use as a benchmark, followed by the distributed algorithm and the improvements of it. Finally we present the results from an experiment in a real environment and discuss the performance of the algorithms.

4.1 Common methods

The UWB localization network has to be determined prior to any tag localization. The network determination consists of several tasks, for example defining the area to provide with localization and placing the anchors. Provided that we already posses the functional hardware and software for the positioning, the network building process can be summarized with several steps:

1. Investigate the area of interest and specify target positioning accuracy

2. Define the coordinate system

3. Suggest the placement of infrastructure and make a Dilution of Precision (DOP) analysis
    - If the DOP values are not satisfactory (too high where it should be low), change the placement in that area so that DOP lowers to a tolerable level and repeat 3
4. Obtain precise positions of anchors
5. Load the necessary data into the infrastructure nodes
6. Place the anchors to given places and make necessary interconnections

Before going to the merit of the network definition, we will briefly describe the points of the plan above.

Objective of the first step it is to define the accuracy goal. It is necessary to decide in which parts of the area it is required to have better performance in terms of lower estimation error, where it is acceptable to have a higher error and where it is sufficient to detect the presence of a tag only. Results of this step highly influence the total number of anchors as lowering the positioning error and irregularity of space require an increase of the number of anchors.

The second of the proposed steps consists of choosing a coordinate system and its origin. For an indoor localization any xyz Cartesian, preferably right-handed, coordinate system is suitable. For example the East-North-Up (ENU) system, where x-axis is pointing towards the east, y-axis to the north and z-axis upwards, is a good choice. In some cases aligning the horizontal axes with the perpendicular building walls is preferred. The exact position of origin is arbitrary. At this point we can also obtain the GNSS coordinates of the coordinate system origin and determine the transformation between the local system and the GNSS coordinate system. With this transformation we are able to convert the estimated local positions of the tags to the global coordinate systems, i.e. perform georeferencing, and for example visualize them on a map.

In the third step we try to place the anchors so that the covered area is maximized while trying to keep the number of them as low as possible (to keep the costs and number of connections low). Then, the anchors will inherently be divided into separate groups, divided by walls or other obstacles. We must then ensure that there always exists a link between the groups in a form of at least one anchor pair, one from each of the neighboring groups, with a line of sight between them. Anchors from such pairs are suitable candidates for the relay anchors. If the groups are connected in this fashion, we should end up with a connected localization network (connected graph).

In order to evaluate the theoretical performance of the network it is useful to perform a DOP analysis. The calculation of the DOP parameters is in Appendix A. Results of the analysis helps to find the areas in a network where the localization error could be above the tolerable level. If there is such an area, we should rearrange (or add) the anchors while preserving the network connectivity. This step is repeated until all the demands on the network are satisfied. It should be noted that this step aims to find a compromise between the cost (number of anchors) and performance. When the infrastructure placement is final, a master anchor and relays should be chosen.

These first three steps are simple in comparison to the fourth step, because they can be done rather quickly, using only a simple floor plan (although detailed plan is better for the DOP analysis) and do not require physical presence of an engineer in the area.

The fourth step, the precise survey of the anchor locations, is the most challenging one. It is necessary for the correct function of the positioning algorithm to know the positions of the anchors as accurately as possible. Otherwise the algorithm would not know how to interpret the measured data as there would be too many unknown variables to estimate. Moreover, for the purpose of anchor synchronization we need to know the distances between the master (or relay) and slave anchors.

Until recently this task has been done manually using a simple laser distance measures, more advanced measuring system or total stations with various degree of accuracy. However, this proves to be very time consuming. It can also become very costly if a high precision of anchor positions (and thus localization) is needed because it requires more people to be involved and more expensive equipment.

Difficulties connected with the manual measurement could quickly outweigh the benefits of the localization, especially in cases when the network is of medium size or the network is needed to be operational in a matter of minutes.

Implementing a self-localizing semi-automated protocol into anchors is therefore a very interesting and desirable option. Since the implementation of the TWR protocol would enable the creation of a functional network on demand we will call such network an ad hoc UWB network. Because only the UWB networks are considered, we will omit the UWB part and refer to the network simply as the ad hoc network. Also, within this section, we will refer to the protocol or algorithm that estimates the anchor positions simply as the algorithm.

The main goal for the ad hoc network creation is to measure a *sufficient* number of anchor-anchor distances and have them as inputs to the algorithm that estimates the anchor positions. The word sufficient hides behind itself a process of anchor pair selection for the distance measurements. Naturally, there are a few requirements on the set of pairs that have to be satisfied, so that the anchor positions could be estimated:

- When estimating the positions in two dimensions, the pairs should be selected in a way that every anchor is present in *at least three* different pairs, giving us at least three distances. The reason for that number comes from the used localization method for position estimation (see Section 1.2.1). The position is estimated by finding the intersection of circles, where we need three circles (three distances) to have only one point of intersection (an unique solution). Should the requirement be lowered to only two distances, then we are most likely to obtain an ambiguous solution[14] that cannot be solved without additional constraints.

  For three-dimensional estimation the number of distances rises to four (intersecting four spheres). Nonetheless, the observability of the height component is typically marginal, since the anchors are usually organized in a horizontal plane. The height can also be measured rather easily and accurately without a specialized equipment, in comparison to the horizontal coordinates.

- Distance between an anchor pair should be measured only if there is a clear line of sight between them.

The anchor determination algorithm also needs initial conditions which can be either a few (two to four) known anchor positions (the ones that can be easily measured) or set of rules that help to define the coordinate system and select correct solution from non-unique ones (e.g. anchor $A2$ is left of anchor $A1$, anchor $A3$ is at origin $\mathcal{O}$).

[14]In such case, one solution is a mirrored version of the other.

The algorithm, proposed by [32], views the anchor network as a graph. In its nature, the algorithm is a *graph algorithm* which operates with the nodes of the graph, the anchors, and which is designed for distributed operation. Before we proceed further, we have to define several terms concerning graphs and graph theory and how they are related to the anchors and UWB network.

Utilization of graph theory

A usual definition of a graph $G = G(V, E)$ is that it is an ordered pair of two sets, where V is a set of graph vertices or nodes and E is the set of edges, connecting the vertices [33]. A subgraph $S = S(V_s, E_s)$ of a graph G is then a graph, whose set of vertices V_s and edges E_s are subsets of the sets V and E.

$$G = G(V, E), S = S(V_s, E_s), V_s \subseteq V, E_s \subseteq E \tag{4.1}$$

Several properties are assigned to each vertex. However, in our case we are particularly interested in only one property, which is the degree of a vertex. A vertex degree $d(v)$ of a vertex v is a number of edges that enter or leave the vertex. We can further divide the degree into in-degree $d_i(v)$ (number of incoming edges) and out-degree $d_o(v)$ (number of outgoing edges) of a vertex v, while the equality $d(v) = d_i(v) + d_o(v)$ holds.

It is possible to define two graphs on the anchor network. First, the visibility graph $G_v = G_v(V_v, E_v)$, where the V_v is a set of all anchors within the network, and E_v is a set of visibility edges. An edge $e_{ij} = (i, j) \in E_v$ exists between anchors i and j if there exists a clear line of sight between them. Second, we define the distance measurement graph $G_d = G_d(V_d, E_d)$, where the V_d is again a set of anchors and E_d is a set of measurement edges. An edge $e_{ij} = (i, j) \in E_d$ between anchors i and j exists if a distance between them has been measured.

These two graphs generally share the set of vertices $V_d = V_v$, while the distance edge set is a subset of the visibility set $E_d \subseteq E_v$, making the distance graph a subgraph of visibility graph.

4.2 Problem definition and established solutions

The determination of anchors is a task of finding their individual three-dimensional positions $\boldsymbol{r}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^{\mathrm{T}}$ given the measured distances $d_{i,j}$ and initial conditions either in the form of fixed positions $\boldsymbol{r}_{fi}$ or constraints $x_i > 0$.

The problem is nonlinear, as the positions are estimated using multilateration. We can formulate it as a *nonlinear least-squares* problem. Our goal is to estimate the unknown positions of n_u anchors $\boldsymbol{r}_i$ using the measured distances. The unknown positions are organized into the vector $\boldsymbol{x}$ of unknown variables

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}_1^{\mathrm{T}} & \cdots & \boldsymbol{r}_{n_u}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} . \tag{4.2}$$

The usual formulation is the minimization of the sum of squared differences of estimated distances and the measured ones

$$\min_{\boldsymbol{x}} f_1(\boldsymbol{x}) = \min_{\boldsymbol{x}} \sum_{(i,j) \in E_d} \left(\| \boldsymbol{r}_i - \boldsymbol{r}_j \| - d_{i,j} \right)^2 , \tag{4.3}$$

where $d_{i,j}$ is the measured distance between anchors i and j, E_d is the set of distance edges and $\|\boldsymbol{v}\| = \sqrt{\sum v_i^2}$ is the two-norm of a vector[15].

However, the authors of [32] use different cost function, where also the estimated and computed distances are squared.

$$\min_{\boldsymbol{x}} f_2(\boldsymbol{x}) = \min_{\boldsymbol{x}} \sum_{(i,j)\in E_d} \left(\|\boldsymbol{r}_i - \boldsymbol{r}_j\|^2 - d_{i,j}^2 \right)^2 \tag{4.4}$$

The reason for that is left unexplained by the authors nor is it explained whether the cost function f_2 has the same set of optimal values as the f_1 has. However, neither of the cost functions is convex[16], thus there is no guarantee that any of the algorithms will find the global minimum. Therefore the results of the estimation will be most probably suboptimal, regardless of the chosen cost function.

The reasons for the change might be of both practical and numerical nature. The practical is that for the evaluation of the f_2 we do not have to calculate any square roots, an operation which is slow in comparison with multiplication. The numerical reason is that the function f_2 is continuous, which means that its first derivation is real and defined on the whole function domain, whereas the first derivation of the f_1 is undefined whenever $\boldsymbol{r}_i = \boldsymbol{r}_j$ (results in division by zero).

We assume that the problem will be solved in the anchors, which is an embedded device with limited computing power. Therefore, given the reasons above we will use the cost function f_2 as it is more suitable for our use.

Now with the cost function $f = f_2$ determined, we can define the vector of residuals $\boldsymbol{g}$. For the lth element of vector $\boldsymbol{g}$ it holds

$$g_l(\boldsymbol{x}) = \|\boldsymbol{r}_i - \boldsymbol{r}_j\|^2 - d_{i,j}^2, (i,j) \in E_d \,. \tag{4.5}$$

The cost function f can then be expressed using the vector $\boldsymbol{g}$

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \min_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}} \cdot \boldsymbol{g}(\boldsymbol{x}) \,. \tag{4.6}$$

Koppanyi et. al. [32] propose several algorithms, which solve the estimation problem using distributed calculations, using only the distances and provided initial conditions. The article views the anchor network as a graph and uses graph theory techniques for the estimation. The proposed algorithms are the Consensus Subgradient (CSG) and Accelerated Weighted Gradient (AWG). In every iteration of a distributed algorithm each anchor calculates a part of the solution and shares it with its neighbors using the available communication channel.

Algorithms CSG and AWG have different requirements on the locality of available information. While CSG relies only on the information gathered from the anchor's intermediate neighborhood, AWG algorithm needs to know the graph Laplacian[17], which is considered to be a global information. It could be difficult to correctly and promptly propagate a global information throughout the whole network, because the actual network topology might not be known to us at the time when the estimation takes place.

For this reason we will focus on the CSG algorithm as it uses purely local information. In the following subsections we will first implement a global solution using the Levenberg-Marquardt algorithm. Then the equations of the CSG algorithm from [32] will be stated and

[15]In this chapter we will omit the norm subscript as we will consider only the two-norm.

[16]Besides other important properties, for the convex functions it holds that any local minimum is also a global minimum.

[17]A matrix characteristic to a network, reflecting the connections between vertices.

its principle described. Next, several improvements of the CSG algorithm will be suggested and finally in Section 4.5 we will show the results from a real-world experiment and discuss them.

4.3 Global approach with LM method

The LM optimization algorithm serves as a benchmark to the CSG algorithm and its variants. We expect that this estimator will converge rather fast with more accurate results in comparison with the distributed algorithm, as it has a global information available.

We will only derive the problem equations for the Levenberg-Marquardt algorithm, as this method has already been described in Section 2.3.

We begin by reminding the LM update equation without the measurement weighting

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left(\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k + \lambda_k \operatorname{diag}(\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k)\right)^{-1} \mathbf{G}_k^{\mathrm{T}}\boldsymbol{g}_k \,, \mathbf{G}_k = \left.\frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k} , \tag{4.7}$$

where $\boldsymbol{x}_k$ is kth estimate of the unknown variables, λ_k is the method parameter, $\boldsymbol{g}_k = \boldsymbol{g}(\boldsymbol{x}_k)$ is the residual vector of the cost function and $\mathbf{G}_k$ is its Jacobi matrix at point $\boldsymbol{x}_k$.

Let us suppose that in an ad hoc UWB network there are in total n anchors, where n_k of them have known location and n_u have unknown location. Then the vector $\boldsymbol{x}$ consists of n_u unknown positions.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}_1^{\mathrm{T}} & \cdots & \boldsymbol{r}_{n_u}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{4.8}$$

The residual vector is calculated according to (4.5). For an lth measurement between the anchors i and j with measured distance $d_{i,j}$, the derivative of the corresponding element g_l is

$$\frac{\mathrm{d}g_l(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} = \left[\frac{\partial g_l(\boldsymbol{x})}{\partial \boldsymbol{r}_1}, \quad \cdots \quad, \frac{\partial g_l(\boldsymbol{x})}{\partial \boldsymbol{r}_i}, \quad \cdots \quad, \frac{\partial g_l(\boldsymbol{x})}{\partial \boldsymbol{r}_j}, \quad \cdots \quad, \frac{\partial g_l(\boldsymbol{x})}{\partial \boldsymbol{r}_{n_u}}\right] . \tag{4.9}$$

We can see that most of the derivative elements will be zero, except for those belonging to the unknown anchors from the measurement l. Those elements are equal to

$$\frac{\partial g_l(\boldsymbol{x})}{\partial \boldsymbol{r}_i} = 2(\boldsymbol{r}_i - \boldsymbol{r}_j)^{\mathrm{T}} \,, \frac{\partial g_l(\boldsymbol{x})}{\partial \boldsymbol{r}_j} = 2(\boldsymbol{r}_j - \boldsymbol{r}_i)^{\mathrm{T}} \,. \tag{4.10}$$

The derivatives for each measurement are then combined into the Jacobi matrix $\mathbf{G}$

$$\mathbf{G}(\boldsymbol{x}_k) = \left.\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k} = \left.\begin{bmatrix} \frac{\mathrm{d}g_1(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} \\ \vdots \\ \frac{\mathrm{d}g_m(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} \end{bmatrix}\right|_{\boldsymbol{x}=\boldsymbol{x}_k} . \tag{4.11}$$

4.4 Distributed methods

When creating an ad hoc UWB network it is often not possible or desirable to have every anchor within the network connected to one another and to the computing node via a wired connection (e.g., Ethernet). Without the interconnections it becomes difficult to efficiently and reliably control the network and share data within it.

This denies us to use an algorithm, such as the LM, which would solve the problem at once. For that we would have to collect the measured distances from the anchors to a single point, where the estimation would take place.

Instead we could solve the problem using a distributed algorithm. One of them is presented by [34] and [35] and its utilization for the ad hoc UWB network is described in [32].

The algorithm is called a Consensus Subgradient and it combines the properties of graph algorithms and (sub) gradient optimization methods. In this section we will describe the algorithm and propose several improvements.

4.4.1 Consensus Subgradient algorithm

We begin by stating the cost function

$$\min_{\boldsymbol{x}} \sum_{(i,j)\in E_d} \left(\|\boldsymbol{r}_i - \boldsymbol{r}_j\|^2 - d_{i,j}^2 \right)^2 = \min_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}} \cdot \boldsymbol{g}(\boldsymbol{x}) \,, \tag{4.12}$$

where the $\boldsymbol{r}_i$ are anchor positions, $d_{i,j}$ is a measured distance between anchors i and j and E_d is a set of edges, representing the anchor pairs with measured distance.

The estimated vector $\boldsymbol{x}$ is a vector of all anchor positions, both known (fixed) and unknown. Consequently, a part of the variable vector $\boldsymbol{x}$ is actually known, but with this inconsistency the algorithm implementation becomes simpler. Thus, it is necessary to assure, that the known parts of the $\boldsymbol{x}$ remain fixed through the whole estimation process.

Instead of solving the problem globally at once, like we did in Section 4.3, we try to solve it in distributed fashion. Every anchor i has its own estimation vector $\boldsymbol{x}_k^{[i]}$, containing the estimates of all the positions, and solves the problem using its local information (i.e., the information obtainable from its immediate neighborhood). In our case, we consider the distances to the neighbors and data (estimate) updates from them. The local information is used to calculate the estimate update $\boldsymbol{u}_0^{[i]}$ from $\boldsymbol{x}_k^{[i]}$, which is then shared with the neighbors. The local anchor also receives updates from its neighbors, naturally. All the received update vectors are combined into a new estimate $\boldsymbol{x}_{k+1}^{[i]}$. These steps are done in parallel in all anchors and are repeated until the convergence is achieved. In the terminology of distributed calculations the anchors are called agents.

Every iteration of the CSG algorithm begins with anchor performing a *subgradient iteration* with its current estimate $\boldsymbol{x}_k^{[i]}$ in order to obtain an updated estimate $\boldsymbol{u}_0^{[i]}$ [32].

$$\boldsymbol{u}_0^{[i]} = \boldsymbol{x}_k^{[i]} - \alpha_k \, \boldsymbol{v}^{[i]}(\boldsymbol{x}_k^{[i]}), \quad \text{where} \quad \boldsymbol{v}^{[i]}(\boldsymbol{x}_k^{[i]}) = (\mathbf{G}^{[i]}(\boldsymbol{x_k}))^{\mathrm{T}} \cdot \boldsymbol{g}^{[i]}(\boldsymbol{x_k}) \tag{4.13}$$

The vector $\boldsymbol{g}^{[i]}$ is the residual vector containing only those elements connected with the measurements available in the node i and $\mathbf{G}^{[i]}$ is Jacobi matrix of the local residual $\boldsymbol{g}^{[i]}$.

From the above equation we can observe that the subgradient update is very similar to the gradient descent update, equations (2.39) and (4.14), where the current estimate $\boldsymbol{x}_k^{[i]}$ is updated by a vector $\boldsymbol{v}^{[i]}$ scaled with current step size α_k.

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \, \nabla f(\boldsymbol{x}_k), \; \nabla f(\boldsymbol{x}) = \mathbf{G}^{\mathrm{T}}(\boldsymbol{x}) \cdot \boldsymbol{g}(\boldsymbol{x}) \tag{4.14}$$

The difference is in the step direction vector $\boldsymbol{v}$. While in the gradient descent method it was the gradient of the cost function ∇f, in the subgradient method it is a subgradient of the cost function. While the subgradient $\boldsymbol{v}$ being calculated similarly as the gradient ∇f, both being derivatives of the same objective function, from a mathematical point of view it is different.

Gradient ∇f is calculated using all of the available data and thus really gives the direction of the steepest ascent. On the other hand a subgradient is calculated only with a subset of data, which means that it gives only a partial information about the steepest ascent direction. In other words a subgradient, calculated from a subset of data, can provide only a limited view on how the function changes from the current point if we change only the corresponding subset of variables.

The article [34], provides a definition of a subgradient $\boldsymbol{g}$ of a convex function f at point $\boldsymbol{x}$

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{g}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{x}) \,, \quad \forall \boldsymbol{y} \,. \tag{4.15}$$

It is apparent that a subgradient underestimates the growth of the convex cost function f. In practice we can understand a subgradient as gradient with many elements set to 0. The definition of a subgradient assumes the function to be convex. But our cost function is not convex and therefore the inequality (4.15) does not hold, which means that the subgradient may even overestimate the function growth. Furthermore, it may result in getting the estimates further from what could be a function minimum (either local or global). However, getting a suboptimal solution is a property of the non-convex function minimization, which has to be taken into account while optimizing its value. As we will see in Section 4.5, the estimation results of the CSG algorithm are satisfactory.

The subgradient update (4.13) is performed by every anchor. Once all the anchors finish their computation, the anchors exchange the calculated vectors $\boldsymbol{u}_0^{[i]}$ with each of their neighbors and enter the consensus phase. In graph theory a consensus is when the internal state (in our case the estimates of anchor positions $\boldsymbol{x}^{[i]}$) of every graph node $i \in V$ are equal.

$$\boldsymbol{x}^{[i]} = \boldsymbol{x}^{[j]} \,, \quad \forall i, j \in V_d \tag{4.16}$$

In the consensus part of the algorithm, every anchor combines the received update vectors $\boldsymbol{u}_0^{[j]}$ from the neighboring anchors with its own update vector $\boldsymbol{u}_0^{[i]}$ into the matrix $\mathbf{U}_0^{[i]}$

$$\mathbf{U}_0^{[i]} = \begin{bmatrix} \boldsymbol{u}_0^{[i]} & \boldsymbol{u}_0^{[1]} & \cdots & \boldsymbol{u}_0^{[m_i]} \end{bmatrix}^{\mathrm{T}} \,, \tag{4.17}$$

where m_i is number of neighbors of anchor i and vectors $\boldsymbol{u}_0^{[1]}$ to $\boldsymbol{u}_0^{[m_i]}$ originate from the ith anchor's neighbor.

Here, we point out a change in notation regarding the vectors $\boldsymbol{u}$. Until now have we used subscripts and variable k to denote that the value has been calculated in kth iteration of the CSG algorithm. However, as the consensus part of the CSG is also iterative, we have to also mark the iteration count for the inner consensus iterations. Fortunately the update vectors $\boldsymbol{u}^{[i]}$ are recalculated every CSG iteration so their value is not retained outside the CSG iteration k (this is why we used subscript 0 in equation (4.13)). This allows us to keep the iteration counts separate. We chose the subscript l to denote the iteration number of the consensus phase.

After the construction of the matrix $\mathbf{U}_l^{[i]}$, every anchor makes a consensus iteration. In the iteration a new update vector $\boldsymbol{u}_l^{[i]}$ is calculated by taking a weighted average of matrix's $\mathbf{U}$ rows

$$\boldsymbol{u}_{l+1}^{[i]} = \mathbf{W}_{ii}\boldsymbol{u}_l^{[i]} + \sum_{(i,j)\in E_d} \mathbf{W}_{ij}\boldsymbol{u}_l^{[j]} \,, \tag{4.18}$$

where matrix $\mathbf{W}$ is so called consensus matrix.

After doing the iteration above, the new update vectors $\boldsymbol{u}_{l+1}^{[i]}$ are again shared with the neighboring anchors and then a new consensus iteration is performed (starting with the construction of matrix $\mathbf{U}_l^{[i]}$). The sharing and iterating is repeated until a consensus is achieved, that is until the update vector converges $\boldsymbol{u}_{l+1}^{[i]} \approx \boldsymbol{u}_l^{[i]}$ for all anchors and is equal to a single vector $\boldsymbol{u}$ (so that all $\boldsymbol{u}_l^{[i]}$ are the same for all i). The vector $\boldsymbol{u}^{[i]}$ without the iteration l subscript has the meaning of the converged vector, obtained as a result of consensus iterations.

Once the network achieves the consensus at the end of the consensus phase (4.18), every anchor takes its converged update vector $\boldsymbol{u}^{[i]}$ and uses it as its new estimate of the estimation vector $\boldsymbol{x}^{[i]}$.

$$\boldsymbol{x}_{k+1}^{[i]} = \boldsymbol{u}^{[i]} \tag{4.19}$$

The subgradient update (4.13) and consensus phase (4.18) are repeated until the estimate $\boldsymbol{x}_k^{[i]}$ converges for all anchors.

The matrix $\mathbf{W}$ plays an important role in the convergence of (4.18) and must be chosen wisely. The Metropolis-Hastings definition [35, 36] was chosen by the authors of the article [32] and in this work we will use it as well[18],

$$W_{ij} = \begin{cases} \min\left\{\frac{1}{d(v_i)}, \frac{1}{d(v_j)}\right\}, & i \neq j \wedge (i,j) \in E_d, \\ \sum_k \max\left\{0, \frac{1}{d(v_i)} - \frac{1}{d(v_k)}\right\}, & i = j \wedge (i,k) \in E_d \end{cases} \tag{4.20}$$

where $d(v_i)$ is the degree of ith anchor.

Selected definition uses for weights calculation only the node degrees, which is the number of anchor's neighbors. Advantage of this definition is that it does not require any global information, because the anchor's degree is stored in anchor's memory and can be obtained anytime from any neighbor.

Other possible metrics are discussed in [35] while convergence analysis of (4.18) is provided in [34].

When the CSG algorithm converges to a solution, the vector $\boldsymbol{x}^{[i]}$ will be identical for every anchor and will contain positions of all the anchors. While it seems beneficial that every anchor has the knowledge of the others whereabouts, it implies that every anchor knows the size of the network in advance. This can be viewed as an acceptable global information, as it can be put into the anchors during their placement. The problem is, however, that for large networks it forces the embedded hardware to operate with long vectors and large matrices. Computing matrix products and inverses with large matrices can greatly impact both the computation speed and precision.

It is not necessary for an anchor to know positions of all other anchors, as it is never needed for the positioning (from a single anchor's perspective) or synchronization. Only the positions of the neighboring anchors will be ever needed e.g. for the determination of distance between slave and relay anchors.

The CSG algorithm also requires a significant amount of data to be shared, in the form of the update vectors $\boldsymbol{u}_l^{[i]}$, each iteration during the consensus phase. This vector has the same size as the estimate vector $\boldsymbol{x}$, which means that it will be also large for large networks.

We can see that the CSG algorithm is better suited for smaller networks, where the dimensions of vectors and matrices can still be handled by the embedded hardware. Moreover, the algorithm does not state nor hint how the update vectors should be shared, leaving a crucial part unsolved.

One of our improvements to the algorithm aims to solve both the dimensionality and update vector sharing issues and we will discuss the improvement in the next section.

[18]The definition of $\mathbf{W}$ in the original work [32] is missing a case where $i = j$. Its definition presented in Equation (4.20) is taken from [36].

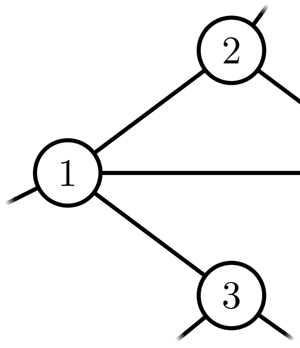


Figure 4.1: Example of an anchor neighborhood

4.4.2 CSG dimension reduction and local consensus

In the previous section we have described the Consensus Subgradient algorithm and discussed its issues, the growth of the estimation vector and the necessary data flow during estimation. In this section we propose several adjustments of the algorithm, which will make it usable even for large networks while preserving the estimation accuracy.

We start with the dimensionality issue. As we have said before, for every anchor it is sufficient to know only the positions of itself and the neighboring anchors. Therefore we propose that every anchor should estimate its own position and the positions of its neighbors only, instead of all the anchors. This greatly reduces the sizes of vectors and matrices and makes the calculations simpler.

With the change of the estimation vector $\boldsymbol{x}^{[i]}$ the update vector $\boldsymbol{u}_l^{[i]}$ and most importantly the construction of matrix $\mathbf{U}_l$ also change. In the original version, the update vector $\boldsymbol{u}_l^{[i]}$ contained positions of all anchors and these update vectors from other anchors were simply combined into the matrix $\mathbf{U}_l^{[i]}$. But with the proposed reduction, the matrix $\mathbf{U}_l^{[i]}$ will contain only the entries corresponding to the neighbors of anchor i. From the neighboring anchors, only the updates concerning the common neighbors can be put into the matrix $\mathbf{U}_l^{[i]}$, meaning that there will be zeros in places of non-common neighbors as not every of the neighboring anchors has the same neighbors as the ith anchor has.

As an example consider the section of an anchor network in Figure 4.1. We will describe the construction of the $\mathbf{U}_l^{[1]}$ matrix from the perspective of anchor 1, which has anchors 2, 3 and 4 as its neighbors. The estimation vector of anchor 1 has the following structure

$$\boldsymbol{x}_k^{[1]} = \begin{bmatrix} \boldsymbol{r}_1^{\mathrm{T}} & \boldsymbol{r}_2^{\mathrm{T}} & \boldsymbol{r}_3^{\mathrm{T}} & \boldsymbol{r}_4^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} . \tag{4.21}$$

When the anchor 1 receives the update vectors from its neighbors, it proceeds to construct the matrix $\mathbf{U}^{[1]}$, where each element $\boldsymbol{u}_{i,j}$ corresponds to the position estimate of ith anchor, received from the jth neighbor (1 being the local anchor 1).

$$\mathbf{U}_l^{[1]} = \begin{bmatrix} \boldsymbol{u}_{1,1} & \boldsymbol{u}_{1,2} & \boldsymbol{u}_{1,3} & \boldsymbol{u}_{1,4} \\ \boldsymbol{u}_{2,1} & \boldsymbol{u}_{2,2} & \mathbf{0} & \boldsymbol{u}_{2,4} \\ \boldsymbol{u}_{3,1} & \mathbf{0} & \boldsymbol{u}_{3,3} & \mathbf{0} \\ \boldsymbol{u}_{4,1} & \boldsymbol{u}_{4,2} & \mathbf{0} & \boldsymbol{u}_{4,4} \end{bmatrix}^{\mathrm{T}} \tag{4.22}$$

The matrix $\mathbf{U}_l^{[1]}$ contains several zero vectors $\mathbf{0}$, that is because several neighbors of the anchor 1 are not common for every neighbor. For example the anchor 3 is a neighbor of anchor 1 but not a neighbor of anchors 2 or 4. Therefore, the anchor 3 is unable to compute the update of position estimates for anchors 2 and 4 as there is no measured distance between them.

If we perform the consensus iterations, with the matrix $\mathbf{U}_l^{[1]}$ constructed as described, the unavoidable presence of the zero vectors in matrix $\mathbf{U}_l^{[1]}$ results in the estimates being pushed towards zero. To cope with this we propose to replace the $\mathbf{0}$ vectors with the parts of the update vector from the local anchor i, which should direct the estimates away from the zero and hopefully towards a solution. The filled matrix from the example would look like

$$\mathbf{U}_l^{[1]} = \begin{bmatrix} \boldsymbol{u}_{1,1} & \boldsymbol{u}_{1,2} & \boldsymbol{u}_{1,3} & \boldsymbol{u}_{1,4} \\ \boldsymbol{u}_{2,1} & \boldsymbol{u}_{2,2} & \boldsymbol{u}_{2,1} & \boldsymbol{u}_{2,4} \\ \boldsymbol{u}_{3,1} & \boldsymbol{u}_{3,1} & \boldsymbol{u}_{3,3} & \boldsymbol{u}_{3,1} \\ \boldsymbol{u}_{4,1} & \boldsymbol{u}_{4,2} & \boldsymbol{u}_{4,1} & \boldsymbol{u}_{4,4} \end{bmatrix}^{\mathrm{T}} . \tag{4.23}$$

This approach has the hazard of moving the estimate away from the local minimum, especially if the local update $\boldsymbol{u}^{[1]}$ is bad. The thorough convergence analysis of this modification is yet to be done. However, we assume that step in the wrong direction will be corrected in the following iteration of the algorithm or that it will result in longer convergence. Nevertheless, it is an acceptable trade off for the size reduction of the local estimate vector $\boldsymbol{x}_k^{[i]}$.

Filling the zero vectors $\mathbf{0}$ in the matrix $\mathbf{U}_l^{[i]}$ with the local updates allows for the second adjustment, which resolves the issue with the large size of necessary data flow. Instead of sharing the update vector $\boldsymbol{u}_l^{[i]}$ after each weighting during a consensus iteration, the anchors should perform only a single consensus iteration (4.18) and share the result as their new estimate $\boldsymbol{x}_{k+1}^{[i]}$. This greatly reduces the data flow and utilization of the communication channel, while also preserving the accuracy of the estimates, which we will prove with experiments within the Section 4.5. We denote this CSG modification as the Neighborhood CSG (N-CSG).

4.4.3 Consensus Levenberg-Marquardt

In addition to the previously mentioned adjustments we have also tried to improve the convergence speed by replacing the subgradient update (4.13) with a different one. We chose to replace it with the basic version of Levenberg-Marquardt update, changing the computation of update vector to

$$\boldsymbol{u}_0^{[i]} = \boldsymbol{x}_k^{[i]} - [(\mathbf{G}_k^{[i]})^{\mathrm{T}}\mathbf{G}_k^{[i]} + \lambda_k \mathbf{I}]^{-1}(\mathbf{G}_k^{[i]})^{\mathrm{T}} \cdot \boldsymbol{g}_k^{[i]} , \tag{4.24}$$

where

$$\boldsymbol{g}_k^{[i]} = \boldsymbol{g}^{[i]}(\boldsymbol{x}_k^{[i]}) , \quad \mathbf{G}_k^{[i]} = \left.\frac{\partial \boldsymbol{g}^{[i]}(\boldsymbol{x}^{[i]})}{\partial \boldsymbol{x}^{[i]}}\right|_{\boldsymbol{x}^{[i]}=\boldsymbol{x}_k^{[i]}} . \tag{4.25}$$

We have named this modification of the CSG as Consensus Levenberg-Marquardt (CLM). We will provide the comparison of how the algorithm's performance changes, when different update methods are used. Note, that the consensus phase of the algorithm will interfere with the LM rules governing the manipulation of the λ_k parameter. Thus, the algorithm may experience sudden increases of the cost function that normally would not happen with the LM method alone.

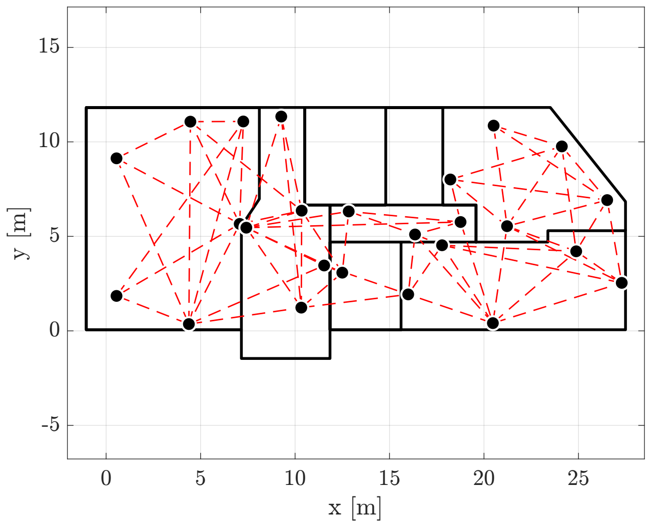


Figure 4.2: Distance graph

4.5 Experiments and comparison

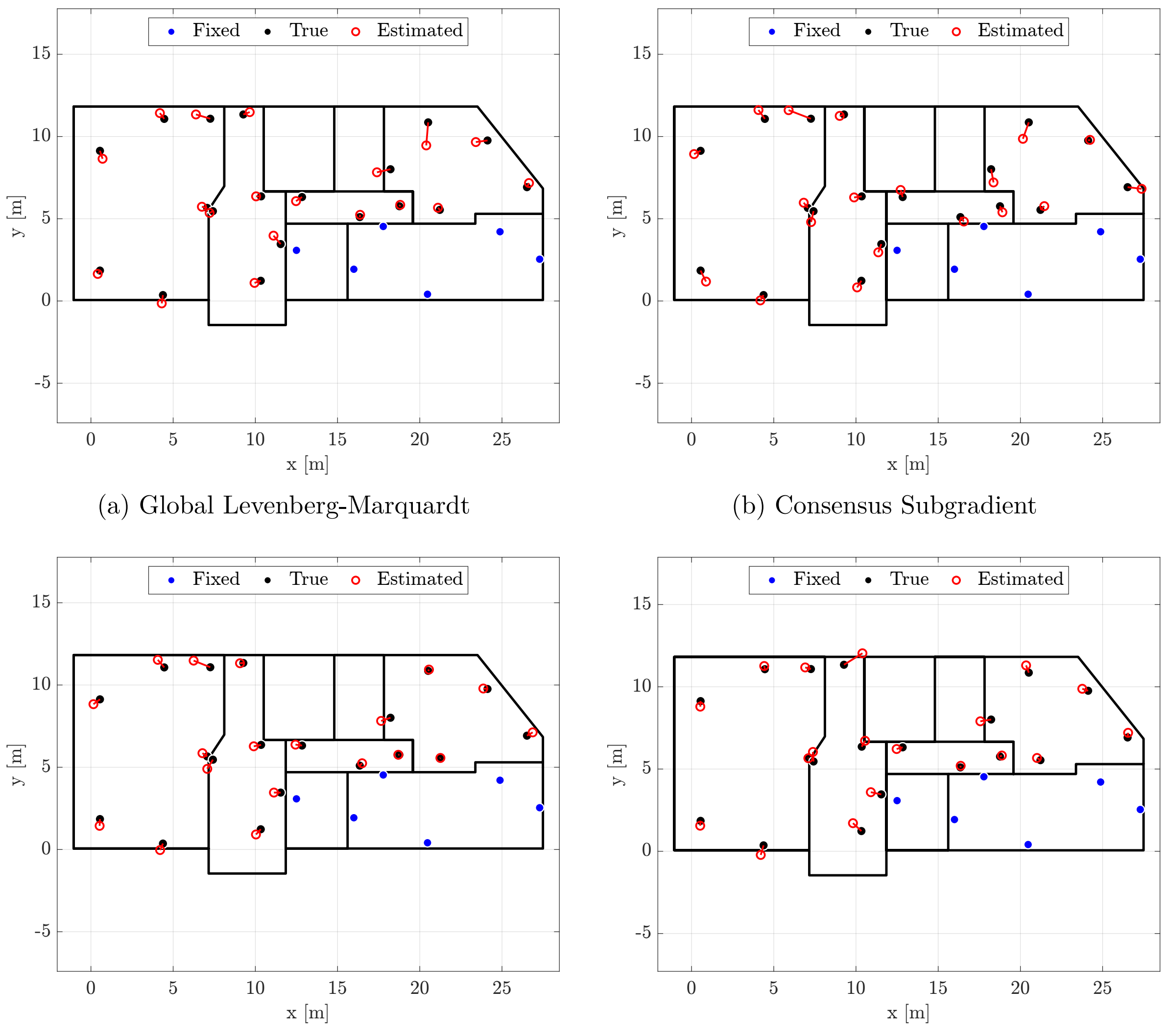
For the evaluation of the ad hoc algorithms we chose to estimate the positions of anchors, which are positioned in the RCD Radiokomunikace headquarters. The UWB network consists of 28 anchors and covers whole floor, across several rooms. The floor plan together with the anchors and distance graph is depicted in Figure 4.2.

For the initial guess we used the true anchor positions perturbed with uniformly distributed random noise in $\pm 2\,\mathrm{m}$ range. By that we simulate the situation when we would guess the anchor positions roughly from a given floor plan. We have selected 5 anchors from the room in the bottom-right corner of Figure 4.2 and one anchor from the room to the left to be used as initial conditions to the algorithms. Given that all the initial anchors are separated from the others by walls, we can expect that there will be a cumulative positioning error, due to the delays and measurement errors caused by the signal propagating through a wall. This error will grow with the distance from the "initial" room. We also expect the error in z-coordinate to be higher than in the horizontal coordinates, because the anchors are mounted under the ceiling, thus having the same height. For the estimation we set the iteration limit to 100 iterations.

The global solution given by the LM algorithm serves as a benchmark. The algorithms will be compared based on their estimation accuracy and convergence speed.

In Figure 4.3 there are the estimation results given by the LM, CSG, CSG estimating only neighboring positions (N-CSG) and the version with LM update (N-CLM). The blue dots denote the fixed initial anchors, black dots are the true positions of the anchors and red circles are the estimated positions. Red line is used to connect the true position with the corresponding estimate.

It is apparent that even with the rough initial guess each algorithm estimated the anchor positions closely to the true positions. However, due to the size of the building Table 4.1 and Figure 4.4 might be more informational on the algorithms performance.



(a) Global Levenberg-Marquardt (b) Consensus Subgradient

(c) Neighborhood Consensus Subgradient (d) Neighborhood Consensus LM

Figure 4.3: Estimated anchor positions

Table 4.1 compares the algorithms based on the estimation error[19]. The columns provide the RMS of the estimation error in x, y and z coordinates, the RMS error in two and three dimensions and the maximal position error.

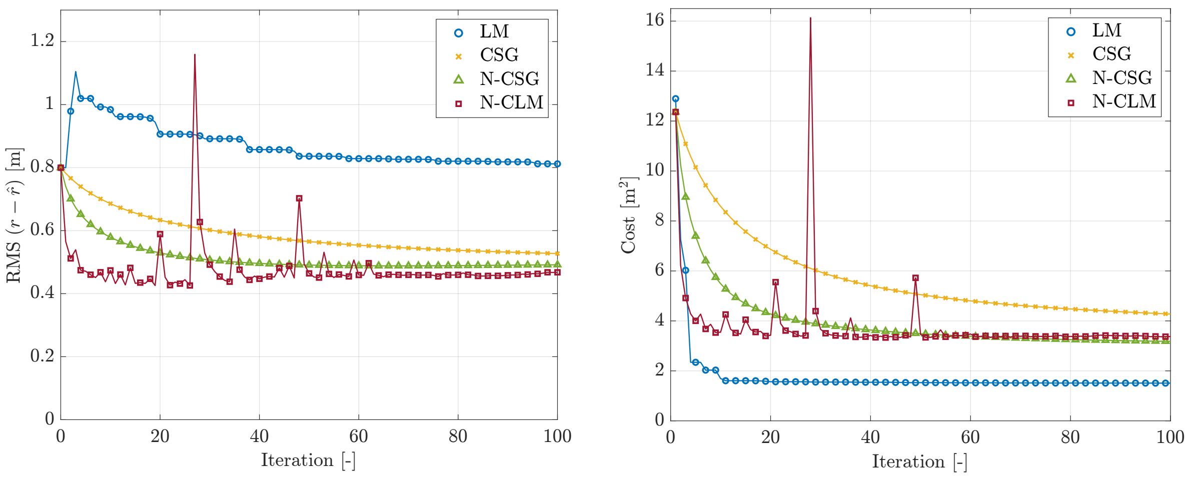
Surprisingly the LM has the worst performance in terms of z-coordinate error, total RMS and maximal error. By looking at the RMS of individual coordinates, it can be seen that the LM achieves low error in horizontal coordinates, while having the worst error in z-coordinate. This is probably caused by the combination of flat anchor geometry and solving the problem as a whole. The flat geometry explanation is supported by the low horizontal RMS and high three-dimensional RMS, which is double in magnitude. The algorithms solving the problem locally perform better in the overall error than the LM algorithm.

The important finding of these experiments is that the N-CSG and N-CLM algorithms, which estimate the position of neighboring anchors only, achieved the lowest estimation errors.

[19]The difference between estimated position $\hat{\boldsymbol{r}}$ and true position $\boldsymbol{r}$.

Table 4.1: Comparison of the final position estimate errors

	RMS [m]			2D	3D	Max
	x	y	z	RMS [m]	RMS [m]	error [m]
LM	0.40	0.41	1.29	0.40	0.81	3.68
CSG	0.45	0.47	0.64	0.46	0.53	1.54
N-CSG	0.38	0.27	0.71	0.33	0.49	1.31
N-CLM	0.39	0.33	0.63	0.36	0.47	1.39



(a) RMS of estimation error (b) Cost

Figure 4.4: Algorithms performance comparison

This is surprising as the N-CSG and N-CLM simplify the problem and in case of N-CLM the LM update interferes with the consensus.

Figure 4.4 provides a view of the estimation error and the current cost in each iteration for all of the used algorithms. Note, that the estimation error may have different behavior than the cost, as the algorithms are controlled by the cost.

In Figure 4.4b we can see that the LM reduces the cost rapidly during the first iterations, however, it keeps to improve the estimate until the last iteration (see Figure 4.4a). We can observe that by the iteration 55 every algorithm, apart from the CSG, has almost reached a minimum and slowed the cost reduction. Although, the N-CSG is still improving its estimate, but rather slowly,

In Figures 4.4a and 4.4b on the plots for the N-CLM we can see that the interference between LM and consensus phase can be an issue. This is caused by the changes to the estimate, done in the consensus phase, are unknown to the LM part of the N-CLM, which results in LM updating the estimate in a wrong direction.

From the results provided in this section, it can be concluded that the CSG, N-CSG and N-CLM algorithms are capable of estimating the global anchor positions given only the local information. For the network used for the experiments, the algorithms can be considered as converged in approximately 55 iterations. However, due to the interfering parts of the N-CLM algorithm, an appropriate cost limit must be set, in order to avoid sudden cost increase.

The interesting findings of the experiments are that even the reduced algorithms N-CSG and N-CLM are able to find a good solution and even outperform the original CSG algorithm.

5 Conclusion

In this work we have continued our development of the position estimation in TDoA-UWB networks. Our past works were concerned with the Tag to Anchor TDoA variant. In this variant the tags are positioned based on their blink messages, which are received by the synchronized anchors. We have recapitulated the key topics of the T2A-TDoA localization in Chapter 2. Also presented were the Chained synchronization algorithm [10] and implementation of the position estimation with the Extended Kalman filter.

One of the two main topics of this work is the positioning with the Anchor to Tag TDoA variant, which we covered in Chapter 3. Here, the anchors broadcast beacon messages that are received by any of the listening tags in range. From these messages, each tag is able to estimate its own position. However, the UWB does not support simultaneous reception of multiple messages, thus the message transmission has to be spaced by a certain delay. Due to the free running clock of the tag devices, the clock drift becomes a dominant source of error, which renders the measurements unusable and which only increases with the increasing message transmission spacing.

The A2T-TDoA method in the UWB networks is not well explored yet and thus, there are only few related sources. Yet, we were able to compensate for the clock drift effects and successfully estimate tag's position. We have achieved this by adjusting the EKF to estimate also the drift dynamics in addition to the position. Moreover, our implementation uses only standardized UWB messages, which allows us to embed data in them, unlike the approaches proposed by [29, 30].

We have evaluated the performance of the approach in a series of static and dynamic tests, some of which are presented in Section 3.3. In the static grid measurement the A2T EKF achieved RMS estimation error of 0.26 m in horizontal plane and error of 0.30 m when estimating in three dimensions. This error would be probably even lower if the anchors were not all mounted in the same height.

Correctly implemented A2T-TDoA allows the user tags to estimate their own position, which makes the real-time navigation possible. Also, there is no limit on the number of concurrently operating tags, since any tag that is able to receive beacon messages is able to locate itself as well.

Furthermore, the UWB network designed in our works is able to operate in both A2T and T2A modes simultaneously. This is a direct result of the introduction of the relay anchors to the network [10] and that the beacon messages are in fact the *relayed* synchronization messages. Such property enables simultaneous tracking (e.g., goods in a warehouse) and navigation (e.g., vehicles, pedestrians, autonomous robots).

Chapter 4 focused on the self-localization of the anchors during the determination of a UWB network. Positions of the anchors within a UWB network are considered to be a fundamental information, without which the estimation of the tag's position would not be possible. Usually a dedicated measuring equipment (laser distance meters, total stations, etc.) is used to determine the positions of the anchors. Such approach is time consuming, especially for large networks, and requires equipment with personnel to operate it.

However, the anchors are able to measure distances between each other using the TWR protocol. To solve the estimation problem we have considered the use of distributed calculations, where every anchor contributes to the problem solving. By contributing, every anchor is provided with the solution (anchor positions) once it is estimated.

In particular, we have focused on the Consensus Subgradient algorithm proposed for this purpose by [32] that uses only the local information (measured distances) for the estimation. The realized experiments indicate that the original CSG algorithm is able to correctly estimate the positions of the anchors. However, it forces every anchor to operate with potentially large vectors and matrices, which could be very time consuming for the embedded hardware. Therefore, we have suggested to reduce the estimation vector of every anchor, so that every anchor estimates only its own position and the positions of the neighboring anchors. We have denoted this modification as the Neighborhood CSG, or N-CSG.

Furthermore, we have suggested to exploit a Levenberg-Marquardt update instead of the subgradient update in the distributed estimation, which could increase convergence speed. Combining it with the previous suggestion we have denoted this algorithm as the Neighborhood Consensus Levenberg-Marquardt, or N-CLM.

The experiments then showed that both N-CSG and N-CLM modifications, in terms of estimation error, performed equally well or even better than the original CSG algorithm. Also, the N-CLM proved to have faster convergence in the experiments. However, the N-CLM suffers from sudden increases of the cost function values, due to the interference with the consensus part of the algorithm. Therefore, there have to be devised additional rules and limits that would stop the N-CLM estimation before the estimate diverges, even temporarily.

Due to the already broad scope of this work, we have omitted two topics, the antenna delay calibration and the power corrections, both having noteworthy impact on achievable accuracy. The effect of signal power on the DW1000 UWB chip time-stamping electronics has been shown and investigated by [9] and [22]. In our future works we would like to explore this topic in detail, particularly with relation to the methods presented within this thesis.

Appendices

A Dilution of Precision

Dilution of precision is a quantitative measure of increase in estimated position covariance given the covariance of input data. A loose definition of DOP is that it is the covariance of the change in position $\boldsymbol{r}$ if we slightly perturb the input data $\boldsymbol{h}$.

$$\mathrm{DOP} = \mathrm{cov}\left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\boldsymbol{h}}\right) \tag{A.1}$$

The DOP is mostly associated with the GNSS [17, 37]. In relation with satellite navigation, the DOP parameters relate the geometry of satellites used for the estimation with the actual position of the user. Generally, if the satellites are covering a large area of the sky, from the user's point of view, then the DOP tends to be low and the estimated positions more precise. An example of constellations that achieve low and high DOP can be seen in Figure A.1.

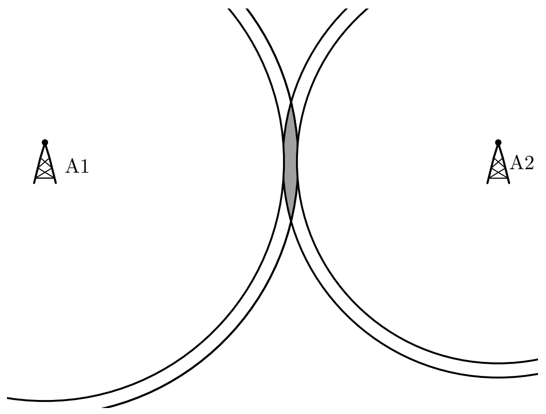
In the context of TDoA, the DOP is related to how the hyperbolae intersect. If the hyperbolae intersect mostly perpendicularly, then the DOP is low. This is depicted in Figure A.2.

Formally, the DOP is calculated using the matrix $\mathbf{G}$ describing the satellite constellation (geometry) w.r.t. user's position. In our case the matrix $\mathbf{G}$ is the Jacobi matrix of the residuals, derived in (2.45) and (2.64). With the matrix $\mathbf{G}$ the relation between the position covariance and covariance of measurement is [37]

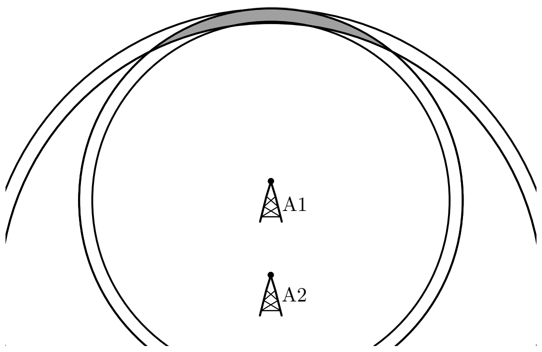
$$\mathrm{cov}(\mathrm{d}\boldsymbol{r}) = (\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}\mathbf{G}^{\mathrm{T}}\mathrm{cov}(\mathrm{d}\boldsymbol{h})\mathbf{G}(\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}\,, \tag{A.2}$$

In case of GNSS the relation above can be reduced to a more compact form, using an assumption of independent measurements and that their variance is identical.

$$\mathrm{cov}(\mathrm{d}\boldsymbol{h}) = \mathbf{I}\,\sigma_{\boldsymbol{h}}^2 \tag{A.3}$$

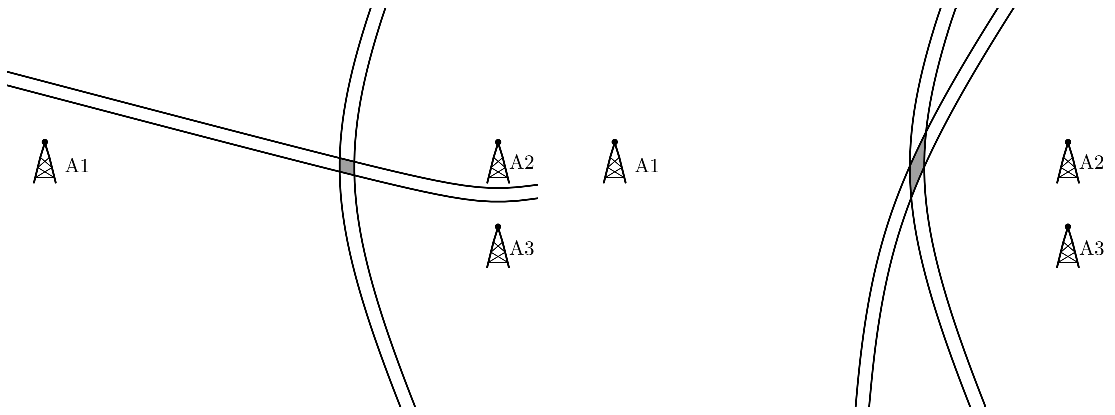


(a) Low HDOP



(b) High HDOP

Figure A.1: Example of HDOP for ToA



(a) Low HDOP (b) High HDOP

Figure A.2: Example of HDOP for TDoA

The assumption of independent measurements does not hold for our TDoA-UWB system, where the covariance can be different for each measurement. However, the relation is useful for the derivation of the DOP. The final relationship of the covariance is then [37]

$$\operatorname{cov}(\mathrm{d}\boldsymbol{r}) = (\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}\mathbf{I}\,\sigma_{\boldsymbol{h}}^{2}\,, \tag{A.4}$$

where the elements of matrix $(\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}$ provide a measure how the measurement covariance and geometry effect the estimation covariance. These elements are used for the calculation of the DOP.

For the estimation of the position in three dimensions, the matrix from (A.4) has the following structure [37]

$$(\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}. \tag{A.5}$$

The matrix $\mathbf{G}$ derived in Section 2.3 can be used in the calculations in (A.5). For the sake of completeness we state the matrix $\mathbf{G}$ for both the TDoA and ToA which is used in the GNSS [37]

$$\mathbf{G}_m^{\mathrm{TDoA}} = \left[(\mathbf{1}_{jr} - \mathbf{1}_{ir})^{\mathrm{T}}\right], \quad \mathbf{G}_m^{\mathrm{ToA}} = \left[\mathbf{1}_{ir}^{\mathrm{T}}\right], \tag{A.6}$$

where m is the number of a measurement, and $\mathbf{1}_{ir}$ is the unit vector pointing from the position estimate $\boldsymbol{r}$ to the ith anchor.

From the a_{ij} elements the vertical, horizontal and positional DOP can be determined as follows

$$\mathrm{VDOP} = \sqrt{a_{33}}\,, \quad \mathrm{HDOP} = \sqrt{a_{11} + a_{22}}\,, \quad \mathrm{PDOP} = \sqrt{a_{11} + a_{12} + a_{13}}\,. \tag{A.7}$$

Each of the VDOP, HDOP and PDOP quantities tell us how much is the estimated position sensitive to the change in data in vertical direction, horizontal direction and in any direction, respectively.

If we return back to the TDoA-UWB systems, the motivation for introducing the height soft constraint in Section 2.3.3 to the estimation, was the bad (high) VDOP of the network. It is often that the anchors are placed to a similar heights, creating a two-dimensional plane. Such network may have good accuracy with estimating the positions in two dimensions (having

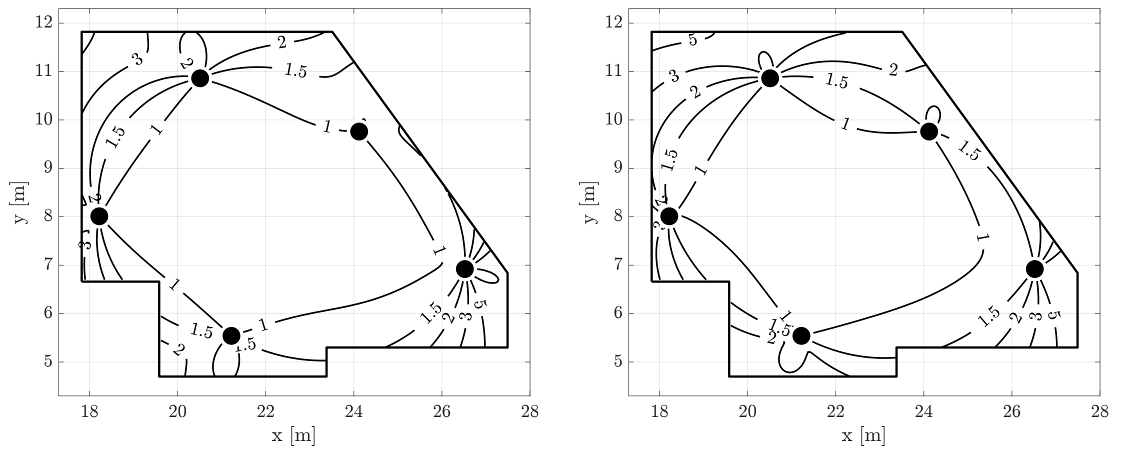
HDOP low nearly everywhere within the network). The network will perform badly when estimating the vertical coordinate, because the hyperboloids will intersect under bad angles, resulting in high VDOP and unreliable estimate of the vertical coordinate.

The DOP parameters can be used for evaluation of the anchor placement withing the network. Providing a tool with which we can determine if the placement is good or if it needs to be adjusted. Another important aspect that influences the DOP in TDoA-UWB networks is the choice of combination matrix **D**. The choice of TDoA pairs determines the hyperboloids that will be intersecting and therefore directly influences the DOP. This is different from the GNSS or ToA systems where the DOP was fully determined by the measurements alone, whereas for the TDoA the DOP is determined by both the measured ToA and their combination into TDoA measurements.

In Figures A.3 and A.4 we can see a real-world example of DOP distribution within a building and how it is affected by the matrix **D**, with the matrices stated in (A.8) and (A.9). In the figures there is a contour plot of HDOP for a room and for a hall, each in two versions that differ in the used combination matrix **D**. The anchors are depicted as the large black circles. Figures A.3a and A.4a use such a combination matrix **D** that the HDOP is good in the whole area. Other Figures A.3b and A.4b use a combination matrix **D** which results in worse HDOP. The most notable difference in the figures is, that the area, where the HDOP is equal or below 1, is noticeably smaller in Figures A.3b and A.4b than it is in Figures A.3a and A.4a.

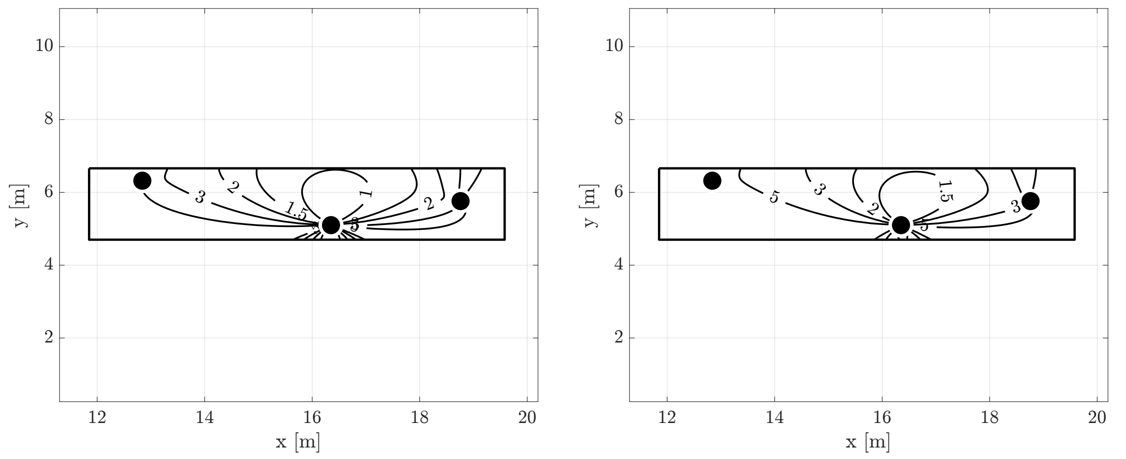
$$\mathbf{D}_1 = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}, \quad \mathbf{D}_2 = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{A.8}$$

$$\mathbf{D}_3 = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}, \quad \mathbf{D}_4 = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \tag{A.9}$$



(a) HDOP with matrix $\mathbf{D}_1$ (b) HDOP with matrix $\mathbf{D}_2$

Figure A.3: Example of HDOP in a room



(a) HDOP with matrix $\mathbf{D}_3$ (b) HDOP with matrix $\mathbf{D}_4$

Figure A.4: Example of HDOP in a hall

B Results of the T2A TDoA

This appendix serves as a recapitulation of key results achieved with the TDoA-UWB system developed in our previous works [3, 4]. The results presented are of three types, each covering a part of the localization, described in Chapter 2. The chosen topics are anchor synchronization and the improvement of adding bias drift rate as a state, position estimation performance comparison using Levenberg-Marquardt method and Extended Kalman filter and finally the effect of adding soft constraint on the tag's height.

For the presentation of the synchronization results we have chosen a test data, where one of the anchors was warming up, shortly after power-on. That is when the clock drift changes rapidly and where the two-state Kalman filter faces major difficulties in synchronizing the anchor. The benefit of estimating the bias drift rate with the three-state KF is best demonstrated on datasets affected by rapidly changing drift.

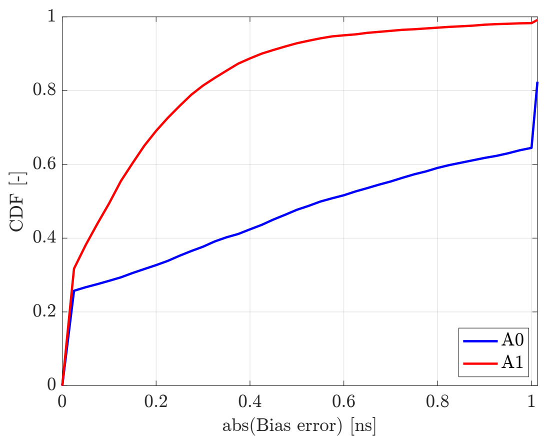
In Figures B.1 and B.2 we can see comparison of the two-state and three-state KF, using the same dataset. The data presented on the figures were generated from the synchronization messages, sent by master anchor, and from the estimates of the Kalman filters. Figures B.1 and B.2 always contain data from two anchors $A0$ and $A1$, where $A0$ was at the warm-up stage during the measurement, whereas the $A1$ was at steady state. Figure B.1 contains a Cumulative Density Function (CDF) of the estimation error of the bias for both of the two anchors. We can see that in Figure B.1a the anchor $A0$ is not able to correctly estimate the bias, due to the unanticipated changes in the bias drift, and more than 35 % of the errors are larger than 1 ns. Such a high error[20] would definitely result in a drop of positioning accuracy. In contrast, in Figure B.1b we can see that the introduction of the bias drift rate to the model helped to compensate the bias drift change and greatly reduce the error (over 95 % lower than 500 ps).

The findings mentioned above are supported by the evolution of KF states, bias drift $\dot{b}$ and bias drift rate $\ddot{b}$, which are plotted in Figure B.2. In Figure B.2a we can see the state vector development or the two-state KF (understandably with the bias drift rate $\ddot{b}$ fixed to 0). In this case, the bias drift of the anchor $A1$ remained constant during the measurement. The bias drift of anchor $A0$ was changing during the first five minutes of the measurement, suggesting that the bias drift rate nonzero. The size of the bias drift rate and the resulting change in the bias drift for the anchor $A0$ must have been higher than the corresponding process noise variance of the filter. Otherwise the error would not be that high for the two-state filter, as seen in Figure B.1a.

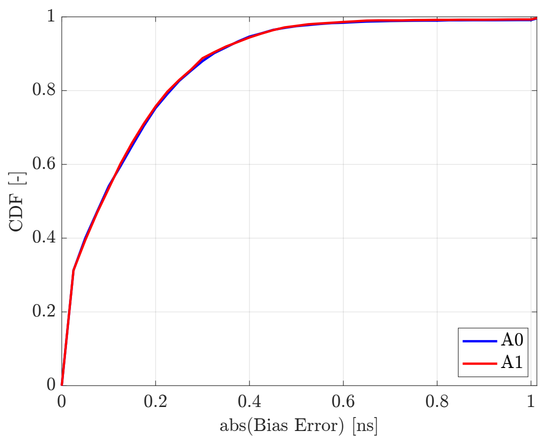
Figure B.2b depicts the states for the three-state KF and it confirms that there has been indeed a change of bias drift, possibly due to the changing temperature of the device, which has over time stabilized near zero (as the temperature stabilized).

By introducing the third state we were able to greatly reduce the synchronization error and use the anchors for positioning even as early as during the warm-up phase. The reduction of the error is summarized by Table B.1, where the listed error is the RMS of the synchronization

[20]The synchronization error of 1 ns is equivalent of approximately 30 cm of ranging error. Given that during positioning several measurements are combined, the error multiplies, resulting a very poor position estimate.

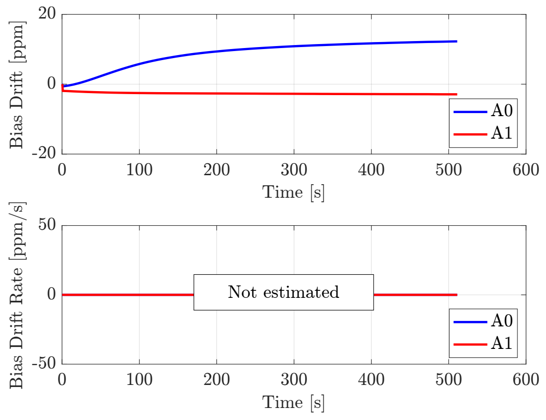


(a) Two-state KF

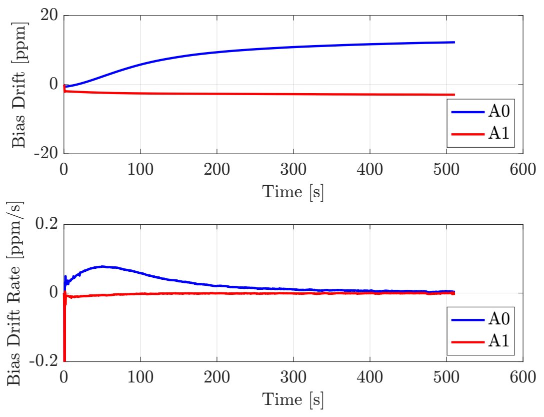


(b) Three-state KF

Figure B.1: CDF of bias error



(a) Two-state KF (drift rate set to zero)



(b) Three-state KF

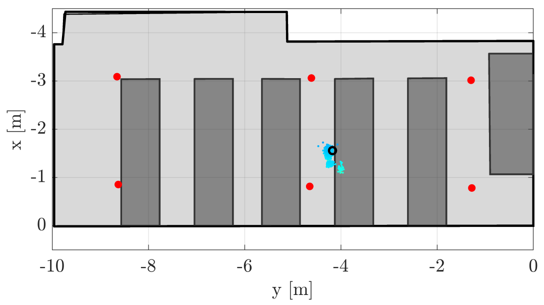
Figure B.2: Time development of bias drift and bias drift rate

error from the whole measurement. It is remarkable that with the proposed three-state KF we are able to achieve very low synchronization error, as low as 300 ps (9 cm).

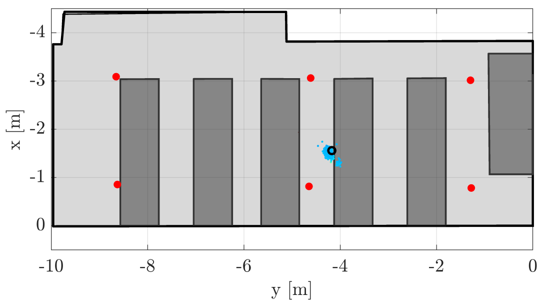
The next topic of the TDoA-UWB system development was about the position estimation from TDoA measurements. Position estimation is a nonlinear problem and we chose to solve it using two approaches. Firstly, we chose to view the problem as a NLSQ one a solve it with epoch-by-epoch approach, using the Levenberg-Marquardt algorithm. For the second approach was used the Extended Kalman filter. Both algorithms achieve great accuracy with

Table B.1: Synchronization RMS error for two and three-state Kalman Filter

	Two-state KF		Three-state KF	
	RMS [ns]	Mean [ns]	RMS [ns]	Mean [ns]
Anchor $A0$	2.302	1.295	0.305	-0.007
Anchor $A1$	0.306	-0.099	0.244	0.002

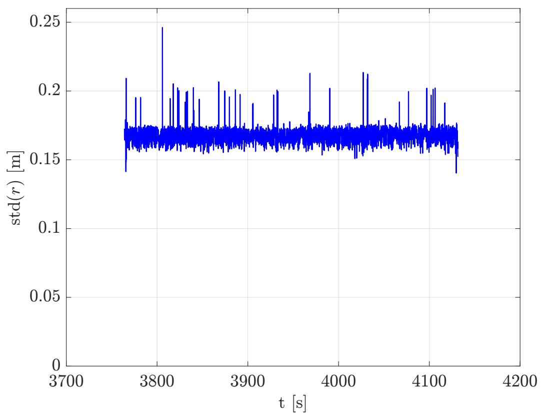


(a) Levenberg-Marquardt

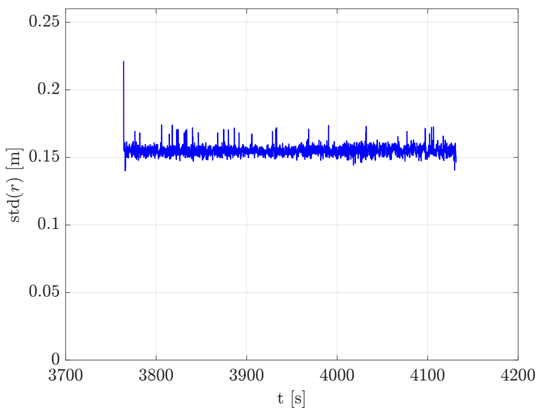


(b) Extended Kalman Filter

Figure B.3: Comparison of LM and EKF based on estimation results



(a) Levenberg-Marquardt



(b) Extended Kalman Filter

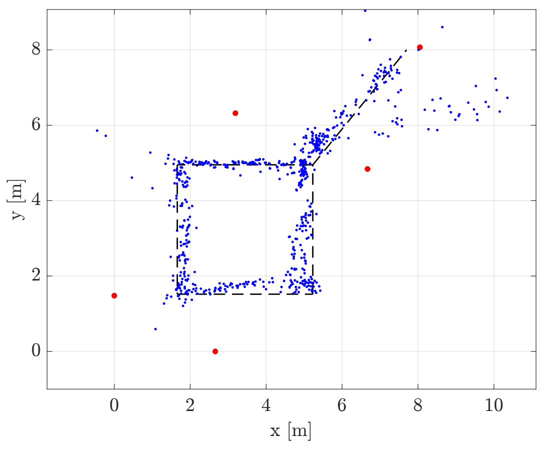
Figure B.4: Comparison of LM and EKF based on total estimation standard deviation

the given data and both have almost equivalent computational costs. We will compare the algorithms by the estimation results and standard deviation of the estimates.

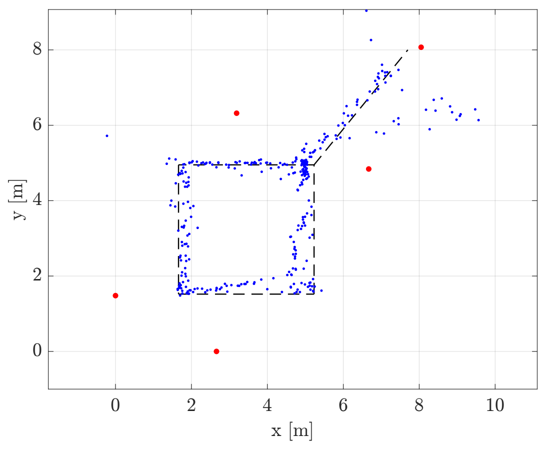
For the comparison we chose data from a static test in a classroom with six anchors and the tag placed on a tripod. So the position of the anchors and the tag is precisely known. In Figure B.3 we can see the resulting estimates from horizontal view. The anchors are plotted as red dots, while the ground truth for the tag's position is depicted as a black circle.

From the estimated positions in Figure B.3 we immediately notice that the estimates of both algorithms are very accurate and that the EKF estimates exhibit lower position variance. Indeed, the standard deviation of the estimates confirms it. The standard deviation has been calculated from the position estimate covariance, which was estimated by the LM and EKF algorithms. The evolution of the standard deviation during the measurement is displayed in Figure B.4.

Finally, we compare the estimation results with and without the use of constrained height. The motivation of using the constraint is to reduce the uncertainty of the estimation of tag's vertical z-coordinate. This is useful when the anchors are in the same height and the vertical uncertainty is substantially higher then the horizontal. If we have the information that the tag will be moving at more or less constant height, we can put this information into the estimation and make the estimates more accurate.



(a) Height unconstrained



(b) Height constrained to 1 m

Figure B.5: Effect of height constraint on estimation results using LM

Even if we are not interested in the z-coordinate, we still might be interested in adding the constraint, because the excessive uncertainty in the z-coordinate can affect the estimation of horizontal coordinates and make it less accurate.

In Figure B.5 we can see the comparison of position estimates with and without the height soft constraint. The estimates were calculated using LM algorithm. The test was done with five anchors and a tag attached to a trolley, making it move at constant height. The anchors were mounted 2.7 m above ground, while the tag's height was 1 m. The anchors are displayed as red dots, tag's positions as blue dots and the ground truth of tag's movement as black dashed line.

We can see that the uncertainty in the z-coordinate may really affect the estimation in horizontal plane as can be seen in Figure B.5a. The estimates in the left part of the figure are far from the true position due to the uncertain height. By implementing the height soft constraint the estimates return closer to the true value, as it is shown in Figure B.5b.

Bibliography

1. P. Davidson and R. Piche, "A survey of selected indoor positioning methods for smartphones", *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1347–1370, 2017. DOI: 10.1109/comst.2016.2637663.
2. "IEEE standard for low-rate wireless networks", *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, 2016-04. DOI: 10.1109/IEEESTD.2016.7460875.
3. J. Krška, "Kalmanova filtrace výstupních dat polohy ze systému UWB a jejich fúze s daty GPS sensoru", Czech Technical University in Prague, 2018.
4. V. Navrátil and J. Krška, "Určování polohy UWB-TDoa: Popis metod, výsledky a doporučení", Centrum integrovaných družicových a pozemských navigačních technologií, Internal report no. TE01020186/CTU/2017/8, 2017.
5. Z. Sahinoglu, S. Gezici, and I. Güvenc, *Ultra-wideband Positioning Systems Theoretical Limits, Ranging Algorithms, and Protocols.* Cambridge University Press, 2008, ISBN: 978-0-511-54105-6.
6. V. Navrátil, "Algoritmy určování polohy a fúze dat pro radionavigační systémy", PhD thesis, Czech Technical University in Prague, 2019.
7. "ETSI EN 302 065-1", European Telecommunications Standards Institute (ETSI), 2016-11, Short Range Devices (SRD) using Ultra Wide Band technology (UWB); Harmonised Standard covering the essential requirements of article 3.2 of the Directive 2014/53/EU; Part 1: Requirements for Generic UWB applications. URL: https://www.etsi.org/deliver/etsi_en/302000_302099/30206501/02.01.01_60/en_30206501v020101p.pdf.
8. "ETSI EN 302 065-2", European Telecommunications Standards Institute (ETSI), 2016-11, Short Range Devices (SRD) using Ultra Wide Band technology (UWB); Harmonised Standard covering the essential requirements of article 3.2 of the Directive 2014/53/EU; Part 2: Requirements for UWB location tracking. URL: https://www.etsi.org/deliver/etsi_en/302000_302099/30206502/02.01.01_60/en_30206502v020101p.pdf.
9. Decawave Ltd, *DW1000 user manual*, V. 2.12, 2017. URL: https://www.decawave.com/sites/default/files/resources/dw1000_user_manual_2.12.pdf.
10. V. Navrátil, J. Krška, F. Vejražka, and V. Koreček, "Chained wireless synchronization algorithm for UWB-TDOA positioning", in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, IEEE, 2018, pp. 149–157. DOI: 10.1109/plans.2018.8373376.
11. V. Navrátil, J. Krška, and F. Vejražka, "Experimental evaluation of chained synchronization for UWB systems", in *Proceedings of the 16th IAIN World Congress 2018 (IAIN2018)*, (2018-11-26), Chiba, Japan, 2018, pp. 137–145.
12. D. Neirynck, E. Luk, and M. Mclaughlin, "An alternative double-sided two-way ranging method", in *2016 13th Workshop on Positioning, Navigation and Communications (WPNC)*, IEEE, 2016-10. DOI: 10.1109/WPNC.2016.7822844.
13. V. Djaja-Josko and J. Kolakowski, "A new method for wireless synchronization and TDOA error reduction in UWB positioning system", in *2016 21st International Conference on Microwave, Radar and Wireless Communications (MIKON)*, IEEE, 2016-05. DOI: 10.1109/mikon.2016.7492077.
14. V. Navrátil and F. Vejražka, "Bias and variance of asymmetric double-sided two-way ranging", *Navigation*, vol. 66, no. 3, pp. 593–602, 2019-08. DOI: 10.1002/navi.321.

15. C. McElroy, D. Neirynck, and M. McLaughlin, “Comparison of wireless clock synchronization algorithms for indoor location systems”, in *2014 IEEE International Conference on Communications Workshops (ICC)*, 2014-06. DOI: 10.1109/ICCW.2014.6881189.

16. S. Kim and J.-W. Chong, “An efficient TDOA-based localization algorithm without synchronization between base stations”, *International Journal of Distributed Sensor Networks*, vol. 11, no. 9, p. 832 351, 2015-01. DOI: 10.1155/2015/832351.

17. P. Misra, *Global Positioning System: signals, measurements, and performance.* Lincoln, Mass: Ganga-Jamuna Press, 2011, ISBN: 9780970954428.

18. B. Parkinson, *The Global Positioning System: Theory and Applications.* Washington, DC: American Institute of Aeronautics and Astronautics, 1996, ISBN: 9781563471063.

19. D. Vecchia, P. Corbalan, T. Istomin, and G. P. Picco, “TALLA: Large-scale TDoA localization with ultra-wideband radios”, in *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, 2019-09. DOI: 10.1109/ipin.2019.8911790.

20. G. Welch and G. Bishop, “An introduction to the kalman filter”, University of North Carolina at Chapel Hill, Los Angeles, CA, USA (August 12-17), Tech. Rep., 2006-07.

21. D. Simon, *Optimal State Estimation.* John Wiley & Sons, 2006-06, 554 pp., ISBN: 0471708585. URL: https://www.ebook.de/de/product/5388550/simon_dan_simon_optimal_state_estimation.html.

22. J. Sidorenko, V. Schatz, N. Scherer-Negenborn, M. Arens, and U. Hugentobler, “DecaWave ultra-wideband warm-up error correction”, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 1, pp. 751–760, 2021-02. DOI: 10.1109/taes.2020.3015323.

23. M. I. A. Lourakis, “A brief description of the levenberg-marquardt algorithm implemened by levmar”, 2005. URL: http://users.ics.forth.gr/~lourakis/levmar/levmar.pdf.

24. K. Levenberg, “A method for the solution of certain non-linear problems in least squares”, *Quarterly of Applied Mathematics*, vol. 2, no. 2, 1944, ISSN: 1552-4485. DOI: 10.1090/qam/10666.

25. D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters”, *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, 1963-06, ISSN: 0368-4245. DOI: 10.1137/0111030.

26. A. N. Tikhonov, “On the stability of inverse problems”, *Proceedings of the USSR Academy of Sciences*, vol. 39, pp. 195–198, 1943.

27. R. Fletcher, “A modified marquardt subroutine for non-linear least squares”, Theoretical Physics Division, Atomic Energy Research Establishment, Tech. Rep., 1971.

28. S. J. Julier and J. K. Uhlmann, “New extension of the kalman filter to nonlinear systems”, in *Signal Processing, Sensor Fusion, and Target Recognition VI*, I. Kadar, Ed., SPIE, 1997-07. DOI: 10.1117/12.280797.

29. P. Corbalán, G. P. Picco, and S. Palipana, “Chorus”, in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, ACM, 2019-04. DOI: 10.1145/3302506.3310395.

30. B. Großwindhager, M. Stocker, M. Rath, C. A. Boano, and K. Römer, “SnapLoc: An ultra-fast UWB-Based indoor localization system for an unlimited number of tags”, in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, ACM, 2019-04. DOI: 10.1145/3302506.3310389.

31. A. Ledergerber, M. Hamer, and R. D’Andrea, “A robot self-localization system using one-way ultra-wideband communication”, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015-09. DOI: 10.1109/iros.2015.7353810.

32. Z. Koppanyi, C. K. Toth, and D. G. Brzezinska, “Scalable ad-hoc UWB network adjustment”, in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, IEEE, 2018-04. DOI: 10.1109/plans.2018.8373544.
33. J. A. Bondy and U. S. R. Murty, *Graph theory with applications*. New York: Elsevier Science Publishing Co. Inc, 1976, ISBN: 0444194517.
34. B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, “Subgradient methods and consensus algorithms for solving convex optimization problems”, in *2008 47th IEEE Conference on Decision and Control*, IEEE, 2008. DOI: 10.1109/cdc.2008.4739339.
35. L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging”, *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004-09. DOI: 10.1016/j.sysconle.2004.02.022.
36. S. Boyd, P. Diaconis, and L. Xiao, “Fastest mixing markov chain on a graph”, *SIAM Review*, vol. 46, no. 4, pp. 667–689, 2004-01. DOI: 10.1137/s0036144503423264.
37. E. D. Kaplan and C. J. Hegarty, *Understanding GPS: Principles and Applications*, 2nd ed. Artech House, 2006, ISBN: 978-1-58053-894-7.

Lists

List of Symbols and Operators

b_i	Clock bias of ith UWB device w.r.t. reference time domain.
c	Speed of light.
$\mathrm{diag}(\boldsymbol{x})$	Operator that creates a diagonal matrix from the input vector.
$\mathrm{Diag}(\mathbf{A})$	Operator that creates a diagonal matrix from the input matrix by clearing the off-diagonal elements.
d	Distance.
$\mathbf{D}$	Combination matrix for TDoA measurements.
$\mathrm{E}[X]$	Expected value of a random variable X.
h	TDoA measurement interpreted as a distance difference.
$\tilde{h}$	TDoA measurement.
$\Phi(m)$	A mapping that maps the TDoA measurement number m to a pair of anchors i and j, which were used to create the measurement.
$\mathbf{Q}$	Process noise covariance matrix.
$\bar{\mathbf{Q}}$	Process noise covariance matrix normalized to one second.
$\boldsymbol{r}$	Position vector.
τ	Signal propagation time.
t^A	Time measured w.r.t. time domain A.
T_x	Time interval.
σ_X^2	Variance of a random variable X.

List of Acronyms

A2T	Anchor to Tag.
ADS	Asymmetric Double Sided.
AWG	Accelerated Weighted Gradient.
BPM	Burst Position Modulation.
BPSK	Binary Phase-Shift Keying.
CDF	Cumulative Density Function.
CLM	Consensus Levenberg-Marquardt.
CRLB	Cramér-Rao Lower Bound.
CSG	Consensus Subgradient.
DOP	Dilution of Precision.
EKF	Extended Kalman Filter.
ENU	East-North-Up.
GNSS	Global Navigation Satellite System.
GPS	Global Positioning System.
GSM	Global System for Mobile Communications.
HDOP	Horizontal Dilution of Precision.
HRP	High Rate Pulse repetition frequency.
KF	Kalman Filter.
LM	Levenberg-Marquardt.
LOS	Line of Sight.
MEMS	Microelectromechanical Systems.
NLSQ	Nonlinear Least-Squares.
PDOP	Positional Dilution of Precision.
PHY	Physical.
PI	Proportional-Integral.

PID	Proportional-Integral-Derivative.
PII	Proportional-Integral-Integral.
PLL	Phase-Locked Loop.
PoE	Power over Ethernet.
ppm	Parts Per Million.
PSD	Power Spectral Density.
RMS	Root Mean Square.
RSSI	Receive Signal Strength Indicator.
SDS	Symmetric Double Sided.
SNR	Signal to Noise Ratio.
T2A	Tag to Anchor.
TDoA	Time Difference of Arrival.
ToA	Time of Arrival.
ToF	Time of Flight.
TWR	Two Way Ranging.
UKF	Unscented Kalman Filter.
USB	Universal Serial Bus.
UTC	Temps Universel Coordonné.
UWB	Ultra Wide Band.
VDOP	Vertical Dilution of Precision.
WPAN	Wireless Personal Area Network.

List of Figures

List of Tables