Lodz University of Technology
Faculty of Electrical, Electronic, Computer and Control Engineering
Institute of Applied Computer Science

MASTER'S THESIS

System for monitoring user position in mobile mixed reality application

(Unofficial translation into English by the author)

(Last update: 23 November 2026)

Piotr Marcińczyk
213381

Supervisor:
Piotr Duch, PhD

Łódź, September 2018

Contents

Nomenclature **4**

Abstract **5**

1. Introduction **6**
1.1. Background 6
1.2. Aims and scope of the thesis 7

2. Existing solutions **8**
2.1. Mixed reality 8
2.2. Position tracking methods 9
2.2.1. Inertial 9
2.2.2. Optical 13
2.2.3. Other methods 19

3. Solutions used **21**
3.1. The DecaWave system 21
3.2. Server application 27
3.3. Mobile application 34

4. Performance tests **41**
4.1. Autopositioning 42
4.2. Static measurements 44
4.3. Dynamic measurements 50
4.4. Subjective tests 54
4.5. Power consumption test 55

5. Conclusions **57**

Bibliography **61**

List of Figures **63**

Nomenclature

AAR	Android Archive
AR	Augmented Reality
AV	Augmented Virtuality
FPS	Frames Per Second
IEEE	Institute of Electrical and Electronics Engineers
IMU	Inertial Measurement Unit
IR	Infrared Radiation
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
LOS	Line-of-sight
MR	Mixed Reality
Ni-MH	Nickel Metal Hydride battery
NLOS	Non-line-of-sight
RR	Real Reality
RTLS	Real-Time Locating System
SPI	Serial Peripheral Interface
SWD	Serial Wire Debug
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
ToF	Time of Flight
TWR	Two Way Ranging
UART	Universal Asynchronous Receiver and Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
UWB	Ultra Wide Band
VR	Virtual Reality

Abstract

The following master thesis describes realisation of a system for monitoring user position in a real world using ultra wide band radio waves and then transforming it to a mixed reality (MR) application running on a mobile device. Position determining is done by **DecaWave TREK1000** kit which allows measurements of the distance between its devices with the precision of a few centimeters by calculating time of flight of radio waves signal (ToF). By using three of them as stationary points of reference and one mounted on the user, the position is calculated using trilateration algorithm, which is then transformed into camera location on the virtual scene. The firmware of these devices has been modified for the needs of this thesis to obtain the best efficiency of ranging. A sample mobile application is created in **Unity** environment for **Android** operating system with **Google VR SDK**. There have been made several tests to check usability in a objective way of the developed solution for mixed reality applications.

Keywords: virtual reality, mixed reality, indoor localization, mobile devices, wireless communication.

Chapter 1

Introduction

1.1. Background

> *"When people take off the headset, they immediately have a creative idea about what they can make in virtual reality, and a lot of them immediately want to get involved."*
>
> — Brendan Iribe, co-founder of Oculus VR[1]

Virtual reality is a rapidly developing field of computer science and electronics. Ever newer solutions keep appearing on the market that let users immerse themselves in a computer-generated world. To make interaction with virtual reality richer, controllers in the form of gloves, cameras that track the user's movements, and even systems that transfer the movements of the whole body into the virtual world are often used. The growing computing power of portable devices such as smartphones and tablets means that they can be used to display the virtual world. Moreover, they often have built-in sensors that determine their position in space, so once placed in a special headset, they can serve as a ready-made way of taking the user into virtual reality.

[1] TechCrunch: *Oculus CEO Brendan Iribe Donates $31M To Build VR Lab At His Alma Mater University Of Maryland*, 12.09.2014, `https://techcrunch.com/2014/09/11/brendan-iribe-center-for-computer-science/` (accessed 05.09.2018)

1.2. Aims and scope of the thesis

The aim of this thesis is to develop an application for mobile devices that generates a virtual world and lets the user move around in it by translating the user's real position into the virtual environment. The position in the real world will be determined with radio waves, using the DecaWave system. To determine the user's orientation in space, the sensors built into the mobile device running the application will be used. The most common of these are the gyroscope, the accelerometer and the compass. Combining the data from these sensors appropriately gives smooth operation, which in turn makes the experience in the virtual world better.

An additional goal is a system that is easy to set up and intuitive to use for a potential end user. The thesis includes an evaluation of the resulting system, possible ways of improving its performance, and alternative applications not directly related to the topic of the thesis.

Chapter 2

Existing solutions

2.1. Mixed reality

There are many systems on the market that track the position and movements of a user in a room and then transfer them into a virtual world[(1)]. They are often divided into virtual reality (VR) and augmented reality (AR). In the former, the image is generated entirely by a computer. Looking through a special headset with headphones, the user is completely cut off from the real world (real reality, RR). Usually each eye receives a separate image, through miniature screens and special lenses, which gives an impression of depth. Position sensors such as a gyroscope, an accelerometer and a compass determine the orientation of the head, which is then translated into movements of the camera in the virtual scene. Movement and interaction are done with gamepad-type controllers: either one held in both hands, or two separate ones, one for each hand. Their movement in space is often also determined by built-in inertial sensors (inertial measurement unit, IMU), and representations of the limbs or cursors are displayed in the virtual scene.

Augmented reality, in turn, focuses on enriching reality with computer-generated objects. Interaction with the resulting world is very limited, and because

(1) Road to VR: *Overview of Positional Tracking Technologies for Virtual Reality*, `https://www.roadtovr.com/overview-of-positional-tracking-technologies-virtual-reality/` (accessed 05.09.2018)

a single camera captures the scene, the image is not stereoscopic. The places where computer models are to appear are marked with special markers, in the form of a pattern printed on a sheet of paper. Alternatively, image tracking algorithms (*optical flow*) are used to determine the orientation of the surroundings, which removes the need for markers; the position of the generated objects is then chosen by the user. An example of such an application is a program that lets the user try out different arrangements of virtual furniture in a real room[(2)].

Both kinds of systems, as well as all the intermediate forms between them, are referred to as mixed reality (MR) [1][2]. One example is augmented virtuality (AV), which is based on principles similar to VR, but often includes elements of the real world that can be interacted with. A diagram showing how the individual realities are ordered is shown in Figure 2.1. The rest of this chapter describes selected mixed reality solutions available on the market, with particular focus on how they handle interaction and how they determine the user's position in the virtual world.

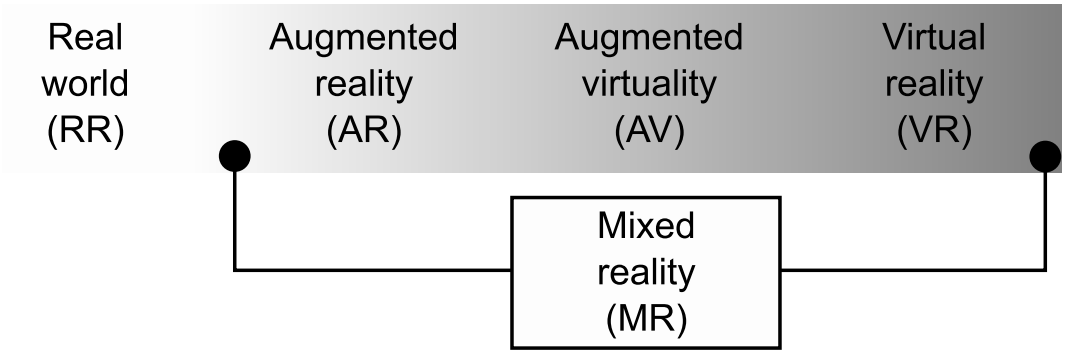


Fig. 2.1: Classification of augmented and virtual realities

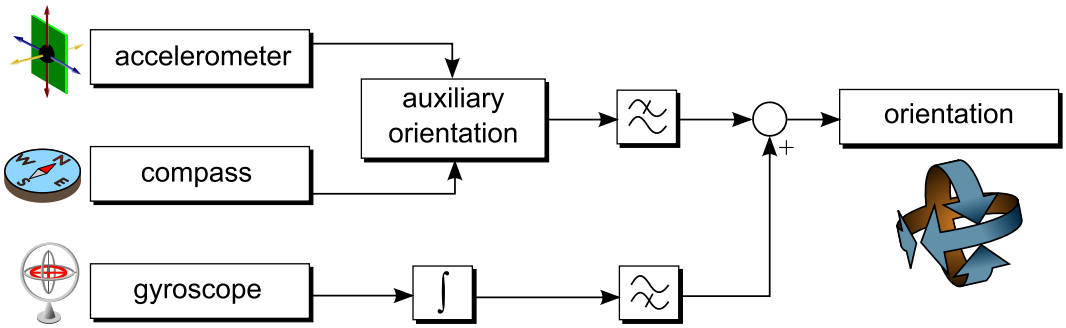
2.2. Position tracking methods

2.2.1. Inertial

As mentioned in the introduction, inertial methods use a set of sensors (IMU) that usually consists of an accelerometer, a gyroscope and a compass. Combining the

[(2)] IKEA Place: `https://www.ikea.com/au/en/apps/IKEAPlace.html` (accessed 05.09.2018)

readings of these sensors appropriately allows the orientation (rotation angle) in space of the tracked device (usually a VR headset) to be determined very precisely. Figure 2.2 shows a typical way of fusing the data from the individual sensors [3][4] to obtain the final result. To simplify somewhat: the measurements from the accelerometer and the compass are used to detect slow changes of orientation. The accelerometer gives the direction of gravity, and the compass the azimuth, i.e. the angle in the horizontal plane; a low-pass filter is then applied to remove noise. The angular velocity readings from the gyroscope are integrated (quadrature) over time, and after the fast changes are extracted, they are combined with the data obtained from the other two sensors in the previous step. This approach minimizes the effects of the accelerometer's high-frequency noise and of the slow drift of the gyroscope, which shows up as a non-zero angular velocity reading at rest.



Source: *P. Lawitzki* [4]

Fig. 2.2: Diagram of IMU sensor data fusion

Inertial orientation tracking is the method most widely used in mixed reality (MR) applications on mobile devices such as smartphones. Most of these devices have the sensors mentioned above, which allow an IMU to be implemented in software, or – more and more often – a ready-made hardware unit that determines the orientation much faster, and thus more smoothly. This has led to rapid development of MR systems based on mobile devices, available to practically everyone. One of the first, and at the same time one of the most popular, is **Google VR SDK**, originally known as **Google Cardboard**[(3)]. Its main idea was to let every smartphone user easily immerse themselves in virtual reality, and to raise developers' interest in

[(3)] Google VR for everyone: `https://developers.google.com/vr/` (accessed 05.09.2018)

this technology. In 2014 an inexpensive viewer was released, made of cardboard, two lenses and a magnet, with instructions for assembling it yourself – shown in Figure 2.3. A phone running the Cardboard app could be placed in it, and holding the



Source: *Wikipedia* [4]

Fig. 2.3: Google Cardboard VR viewer

assembled viewer to the eyes gave access to many interactive 3D simulations and stereoscopic videos. It was controlled with a magnetic button built into the viewer, whose movement was detected by the smartphone's magnetometer. The instructions for building the viewer were published under an open license, so anyone could print and assemble their own at only the cost of the materials. Modified viewers in a wide price range soon appeared on the market, and mobile-based mixed reality developed quickly. In mid-2015 the second version of Cardboard appeared, supporting phones with larger screens and phones running **iOS**, and the magnetic button was replaced by conductive foam imitating the touch of a finger. This made it possible to use the mobile device's compass to determine the orientation in space more accurately.

The next stage in the evolution of this platform is the **Google Daydream** sys-

[4] Wikipedia: *Google-Cardboard.jpg*, `https://en.wikipedia.org/wiki/File:Google-Cardboard.jpg` (accessed 05.09.2018)

tem, intended for selected high-end smartphones[5]. The headset was presented in November 2016 and is made of plastic covered with soft fabric, which can be removed and cleaned. It is controlled with a small wireless remote with its own IMU for detecting orientation, 2 buttons, a volume control and a touchpad, which transfers finger movements to the application and allows interaction with its elements.

Meanwhile, Samsung, together with Oculus, presented its own solution that lets owners of its smartphones use them for mixed reality applications.[6] The **Samsung Gear VR** headset was released at the end of 2015, in November. Unlike other systems of this type, it has a built-in IMU that tracks orientation much more accurately than the phone alone. Gear VR is known for its high build quality, especially of its lenses[7]. Applications are controlled with a wireless controller, shown together with the headset in Figure 2.4. It is powered by two AAA batteries and connects to the phone over Bluetooth. Its functionality is similar to that of the controller supplied with the Google Daydream headset, but it has 4 buttons and a trigger (a button imitating the trigger of a gun) under the index finger. Moreover, many consider its design more ergonomic and intuitive, thanks to the cooperation with Oculus, which has experience in designing MR devices[8].

More advanced IMU systems can also determine position (translation) in a room. Unfortunately, they are susceptible to gyroscope drift and to the noise and measurement uncertainty of the sensors. Determining the position requires integrating (quadrature of) the readings, which, in the case of the accelerometer for example, means that noise accumulates quadratically. As a result, the position diverges more and more from reality over time. Nevertheless, if the errors are frequently corrected with another reference system, the positioning accuracy is usually more than sufficient. For this reason, inertial tracking methods are used to complement other

[5] Google: *Daydream*, `https://vr.google.com/daydream/` (accessed 05.09.2018)
[6] Samsung Gear VR with Controller: *Samsung Gear VR with Controller*, `https://www.samsung.com/global/galaxy/gear-vr/` (accessed 05.09.2018)
[7] VR & AR Wiki: *Samsung Gear VR (2015/2016)*, `https://xinreality.com/wiki/Samsung_Gear_VR_(2015/2016)` (accessed 05.09.2018)
[8] Road to VR: *Gear VR Controller Review*, `https://www.roadtovr.com/samsung-gear-vr-with-controller-review/` (accessed 05.09.2018)



Source: *Samsung* [6]

Fig. 2.4: Samsung Gear VR headset with its controller

solutions that are more accurate than an IMU at determining position over longer periods and larger areas. It is worth noting that none of the mobile mixed reality solutions described above can track both position and orientation with an inertial method.

2.2.2. Optical

After inertial methods, optical methods are the most common way of tracking the user's position in mixed reality. They use video cameras mounted on the device that record static points (*inside-out*), or cameras placed in the room that track the user's movements (*outside-in*).

The former are a natural choice for a large share of mobile augmented reality systems. The camera of a smartphone or tablet tracks markers in the frame, determines its own position from them and generates virtual elements overlaid on the image. This approach can be found in many applications available in app stores. Alternatively, the device can detect feature points automatically and track the movement of the scene from them. An example of such a solution is the "AR effect" add-on to the camera app on Sony Xperia phones[9], which lets the user add animated 3D

[9] Google Play: *AR effect*, `https://play.google.com/store/apps/details?id=com.`

characters to the scene. An example result is shown in Figure 2.5.



Fig. 2.5: Photo taken with the Sony AR effect app

One of the devices using this tracking method is **Microsoft HoloLens**, announced in early 2015 and available as a developer edition a year later [5]. It is a mixed reality (MR) headset that overlays computer-generated graphics on the real view. It needs no external device, because it has a built-in computer running Windows 10. The image is displayed by a pair of special transparent holographic lenses, which let light in from outside and at the same time allow graphics to be overlaid by a miniature projector[10]. Each eye receives a slightly shifted image, which gives an impression of depth. The display technology has the drawback of a limited field of view: 30° horizontally and 17.5° vertically[11]. The position is determined with a wide-angle depth camera, a high-resolution video camera, 4 tracking cameras, 4 microphones, an ambient light sensor and an IMU. The device is controlled by hand gestures recognized by the cameras, and by voice commands through the built-in microphones and the Microsoft Cortana assistant. It has two speakers imitating spatial

`sonymobile.androidapp.cameraaddon.areffect` (accessed 05.09.2018)

[10] The Imaginative Universal: *How Hololens Displays Work*, `http://www.imaginativeuniversal.com/blog/2015/10/18/how-hololens-displays-work/` (accessed 05.09.2018)

[11] VR & AR Wiki: *Microsoft HoloLens*, `https://xinreality.com/wiki/Microsoft_HoloLens` (accessed 05.09.2018)

sound and a 3.5 mm jack for external headphones. The HoloLens headset is shown in Figure 2.6. Alternatively, the **HoloLens Clicker** remote can be used for interaction. It has built-in orientation tracking and a button, which allows more precise control and replaces hand gestures.



Source: *Microsoft* [5]

Fig. 2.6: The Microsoft HoloLens device

Another popular device using *inside-out* optical position tracking is **HTC Vive**, released in April 2016[(12)]. It is a mixed reality headset connected to a computer with an HDMI cable to the graphics card, a USB cable for communication and a power cable. HTC Vive is part of the **SteamVR** system, an extension of the **Steam** platform[(13)] developed by **Valve**, which offers mixed reality games and applications that work not only with this headset but also with similar solutions such as the Oculus Rift or Windows Mixed Reality devices. The latest version, **HTC Vive Pro**, shown in Figure 2.7, has higher-resolution displays of 1440 × 1600 pixels each. In addition, two cameras allow real elements (RR) to be mixed with the simulation. The built-in IMU tracks both orientation and position and is supported by the **Lighthouse** system.[(15)] It uses base stations placed in the room, in the form of boxes, also shown in Figure 2.7. Each of them emits two infrared laser beams, rotating in mutually

[(12)] VIVE: *Discover Virtual Reality Beyond Imagination*, `https://www.vive.com/eu/` (accessed 05.09.2018)
[(13)] *Virtual Reality on Steam*: `https://store.steampowered.com/` (accessed 05.09.2018)
[(14)] Sector: *HTC predstavilo profesionálne balenie Vive Pro headsetu s novými senzormi* `https://www.sector.sk/novinka/149449/htc-predstavilo-profesionalne-balenie-vive-pro-headsetu-s-novymi-senzormi.htm`, 24.04.2018 (accessed 05.09.2018)
[(15)] VR & AR Wiki: *Lighthouse*, `https://xinreality.com/wiki/Lighthouse` (accessed 05.09.2018)



Source: *Sector.sk* [14]

Fig. 2.7: HTC Vive Pro with controllers and Lighthouse base stations

perpendicular planes at about 60 Hz. The signal is received by optical sensors on the headset (located inside its characteristic recesses), which measure the time since a synchronizing flash and use it to determine the angle relative to the station. The position of the headset in the room is then calculated from the known distances between the sensors.[16] In theory, one base station and 5 sensors on the tracked object are enough. For practical reasons, however, two stations and several dozen sensors are used, which increases accuracy and allows devices to be positioned anywhere in a room of up to 5 × 5 m. The manufacturer has also made the positioning system safer to use with the **Chaperone** technology. It lets the user mark the area that is safe to move around in and the positions of obstacles in the room. When the user approaches the boundary they have defined, a clear warning is displayed, with the option of viewing the surroundings through the cameras built into the headset. A further feature of the system is that the simulation or gameplay can be adapted to the available space. Valve decided to publish the technical details and sensors of the Lighthouse system, so that third-party devices supporting this localization method would become more widely available. Thanks to this, the technology can also be used outside mixed reality, for example for positioning drones and robots and for

[16] YouTube: Scott Rumschlag: *Lighthouse Tracking: Basics*, 04.04.2017, `https://www.youtube.com/watch?v=Ikv8o0u6w9o` (accessed 05.09.2018)

precise measurement of rooms.[17][18] Interaction in the HTC Vive system is done with two wireless **SteamVR Controller**s, shown next to the headset in Figure 2.7. They have several buttons, a trigger and a touchpad. Their IMU, together with optical sensors working with the Lighthouse system, tracks their orientation and position in space. The unusual shape of the controllers, ending in an oval extension, comes from the need to place the light sensors where the hand will not cover them, which would prevent the position from being determined correctly. In June 2018 HTC announced the **VIVE Wireless** adapter, which transmits data between the computer and the headset without cables.[19] It also provides 2 to 3 hours of power. Communication is to use the **Wireless Gigabit Alliance** (**WiGig**) technology in the 60 GHz band, with data rates of up to 7 Gb/s. Up to 3 adapters will be able to work in the same room at the same time, which allows multi-user operation.

An example of a system that uses optical tracking from the outside (*outside-in*) is **Oculus Rift**. Its history goes back to 2012, when the company's founder, Palmer Luckey, launched a Kickstarter campaign[20] (Kickstarter is a crowdfunding platform for raising money for projects). The aim was to raise funds to develop and produce about a hundred VR headsets that would be more immersive and intuitive for the average user than those on the market at the time. The project attracted a lot of interest, and in the end about 2.5 million dollars were raised to produce over 7500 developer-edition headsets (DK1).[21] In March 2014 the company was bought by Facebook, and four months later production of the improved DK2 developer edition began. The first consumer version of the headset (CV1) was released at the end of

[17] *SteamVR Tracking*: `https://partner.steamgames.com/vrlicensing` (accessed 05.09.2018)
[18] Road to VR: *Valve Opens Lighthouse VR Tracking API to Third-parties*, 04.08.2016, `https://www.roadtovr.com/valve-third-party-lighthouse-api-steamvr-tracking-htc-vive-dev-kit-license-royalty/` (accessed 05.09.2018)
[19] *VIVE Wireless Adapter*: `https://www.vive.com/us/wireless-adapter/` (accessed 05.09.2018)
[20] Kickstarter: *Oculus Rift: Step Into the Game by Oculus*, `https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game/` (accessed 05.09.2018)
[21] Oculus: *Update on Developer Kit Technology, Shipping Details*, `https://www.oculus.com/blog/update-on-developer-kit-technology-shipping-details/` (accessed 05.09.2018)

March 2016. It is shown in Figure 2.8.



Source: *Heavy Joy Stick* [22]

Fig. 2.8: Oculus Rift with Touch controllers and Constellation cameras

Like the HTC Vive, the headset displays an image from the computer it is connected to. It uses an IMU to determine its orientation, while its position in space is determined with the **Constellation** technology. This consists of infrared (IR) LEDs on the tracked objects and IR cameras that monitor the position of each of them. The light sources are not lit continuously but blink in a specific pattern, transmitting their unique identifier. Knowing how the individual LEDs are arranged on an object, the system can therefore determine its orientation and position in the room. The refresh rate is about 60 Hz; smoothness and precision are improved by also using the IMU to determine the position. The Rift headset has several dozen LEDs on all sides, so it can be tracked at any rotation angle. The simulation can be controlled with the two **Oculus Touch** controllers included in the set, shown in Figure 2.8. Each has two buttons, two triggers – one under the index finger, the other under the middle finger

[22] Heavy Joy Stick: *VirtualLink USB–C Alternate Mode Will Connect VR To PC*, 18.07.2018, `https://heavyjoystick.com/2018/07/18/virtuallink-usb-c-alternate-mode-will-connect-vr-to-pc/` (accessed 05.09.2018)

– and an analog stick. Moreover, capacitive sensors let each button and stick detect whether a finger is resting on it. Orientation and position are determined in the same way as for the headset itself, by combining data from the IMU and the **Constellation** system. The LEDs needed for the latter are on the crescent-shaped extensions at the front of the controllers, hence their code name, *Half Moon*. A single camera is enough to track the headset itself. For the controllers, however, at least two or three are needed, to avoid errors caused by the devices hiding each other. Like the HTC Vive, the system can track objects across a whole room, up to a few meters in each direction.

2.2.3. Other methods

Other position tracking methods include acoustic solutions. Their main idea is to use ultrasound generated by transmitters placed in the room, and microphones on the tracked devices that receive the signals. The simplest variant measures the power of the received signal: the farther the tracked object is from the transmitters, the weaker the sound it receives. This approach is very sensitive to obstruction of the line of sight between the devices, and to noise and interference from other sound sources. Moreover, the receivers must be very sensitive and precise, so this method is rarely used. Much better results can be achieved by measuring the time from sending the signal to receiving it.[(1)] An example of a system that uses this positioning method is **InterSense IS-900**.[(23)] The manufacturer specifies a position accuracy of a few millimeters at a measurement rate of 180 Hz. The system can work both wired and wirelessly (though at the cost of lower accuracy). Its wireless receiver, together with an example pair of glasses (CrystalEyes), is shown in Figure 2.9. One of the system's drawbacks is that it needs a time-consuming calibration before tracking can begin, as well as an additional mechanism to increase accuracy – usually an IMU, as in the **InterSense** technology mentioned above.

[(23)] InterSense: *IS-900 MicroTrax Devices*,
`https://est-kl.com/images/PDF/InterSense/IS-900_MicroTrax_Datasheet.pdf`
(accessed 05.09.2018)
[(24)] Oliver Kreylos' Research and Development Homepage: *KeckCAVES*,
`http://idav.ucdavis.edu/~okreylos/ResDev/KeckCAVES/LinkPictures.html`



Source: *Oliver Kreylos* [24]

Fig. 2.9: InterSense IS-900 device with CrystalEyes glasses

To sum up, it is worth noting that the position tracking methods described in this chapter do not support mixed reality on mobile devices such as smartphones. In particular, dedicated frameworks such as Google VR SDK or Samsung Gear VR offer no official position tracking technology, and third-party solutions of this kind are at the prototype stage. This master's thesis proposes such a system, based on ultra-wideband (UWB) radio communication. The next chapter discusses the technical details of this solution.

(accessed 05.09.2018)

Chapter 3

Solutions used

The system developed in this thesis consists of several parts. Each of them has a clearly defined task, so that together they achieve the goal set out in the introduction. At the very base of the system architecture is the **DecaWave TREK1000** kit, whose task is to determine the user's position with four transceivers. Each of them runs suitably modified firmware, which gives better results in virtual reality applications. The measurements are then sent to a computer running a server application, which processes the data, lets the user change the system configuration and adapts it to the specific environment the user is in. Finally, the position is sent wirelessly to a mobile device, usually a smartphone, which generates the virtual world and, mounted in a virtual reality headset, presents it to the user. Each of these parts is described in detail in the following sections.

3.1. The DecaWave system

DecaWave Ltd is based in Ireland and develops integrated circuits for indoor localization[(1)]. One of its leading products is the **DW1000**, a small transceiver working in the IEEE802.15.4-2011 standard [6]. Its technology allows precise time measurement while data is being transmitted. It can therefore be used in localization systems that also need two-way communication between the transmitters. The po-

[(1)] About DecaWave: `https://www.decawave.com/about` (accessed 05.09.2018)

sition accuracy it provides is about 10 cm, with a range of up to 300 m when the transmitters are in direct line of sight (LOS) and up to 40 m otherwise (non-line-of-sight, NLOS). The chip works in the ultra-wideband (UWB) range, at frequencies from 3 to 6 GHz, with a bandwidth of about 500 MHz. This allows communication with very short pulses (of the order of picoseconds), which makes interference less of a problem than with narrower-band signals. The maximum data rate is 6.8 Mb/s.

The **TREK1000** kit used in this thesis consists of four **EVB1000** evaluation boards that work as transceivers and can communicate with each other. Particularly important is their ability to measure precisely the time from sending a signal to receiving a reply. This makes it possible to determine the distance between the individual transmitters. A single board is shown in Figure 3.1. It carries an **STMicroelectronics STM32F105RCT6** microcontroller, which communicates over an SPI link with the **DW1000** chip described above and processes the received data. The detailed wiring diagram is given in the documentation [11]. In addition, each board comes with an antenna whose parameters were calibrated during production and stored in the microcontroller's non-volatile memory [9]. These include the delay that lengthens the measured time of flight of the signal, and the attenuation that affects the maximum transmit power. The board also has a set of switches for selecting the operating parameters. The first one, labelled `S1`, allows changing, among others:

- the data rate: 110 kbps or 6.8 Mbps,
- the radio channel: 2 (4 GHz) or 5 (6.5 GHz),
- the operating mode: anchor or tag,
- the device address (identifier).

The LCD on the board shows the measured distances and some operating parameters, such as the radio channel and the device address. The device can be powered either from the USB port or from an external source of 3.6 to 5.5 V. More information on configuration can be found in the kit's user manual [7].

The default positioning method of the **TREK1000** kit is called *Two Way Ranging* (TWR) [7]. It consists of two exchanges of data between a fixed *anchor* and a *tag* placed on the tracked object. Both devices must send their reply packets at precisely defined times. This makes it possible to calculate the average time of flight of the

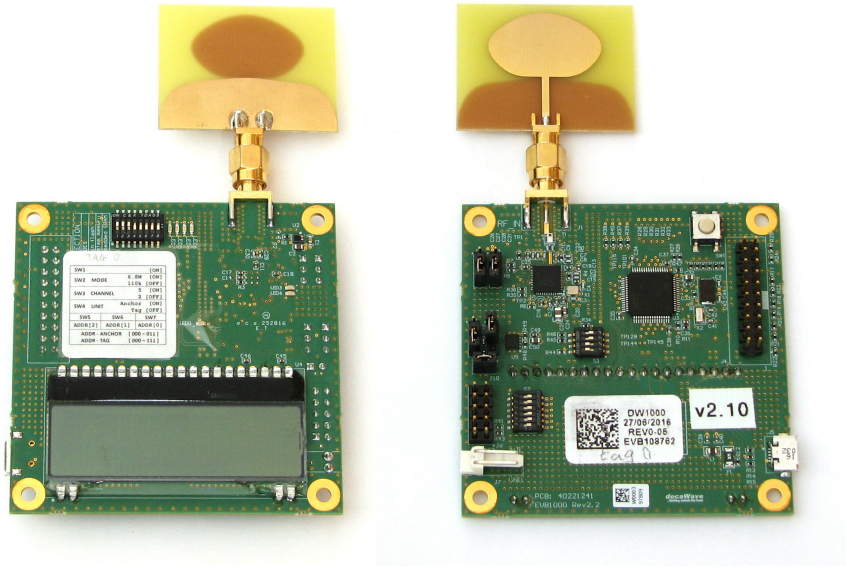


Fig. 3.1: DecaWave EVB1000 evaluation board

signal (ToF), and from it the distance, using the known speed of the signal in air (approximately the speed of light). This approach minimizes measurement errors caused by differences between the clock frequencies of the devices, and removes the need to keep the clocks continuously synchronized by an external source. At least three anchors are needed to determine a position in three-dimensional space. The firmware supplied on the **EVB1000** boards supports communication between up to four anchors and eight tags. A single kit contains only four boards, so in the developed system three of them work as anchors and the fourth as a tag, which is mounted on the user and tracks their position.

The system supports up to eight tags on the same radio frequency. Communication is based on time-division multiplexing (TDM). A full cycle of data exchange between all devices is called a *superframe*. It is divided into 10 time slots: one for each tag and two more for measuring the distances between the anchors, for *autopositioning*. DecaWave optimized the TWR method to reduce the amount of data transmitted and the current consumption. Normally, messages would be exchanged between each pair of devices separately. Since all anchors and tags are within radio range of each other at all times, i.e. there is no *hidden terminal* problem, the number of requests sent can be reduced. The data exchange in a single slot is shown in Fig-

ure 3.2. Communication starts with the tag sending a *broadcast* frame to all anchors in range (*Poll*). Each anchor then replies at a strictly defined moment, in order of device address. The reply frames (*Response*) contain the distance from the given anchor calculated in the previous TWR session. This means that the measurements received are always delayed by the duration of a superframe, which makes the system less effective. At the end of the slot the tag sends a reply (*Final*) containing the timestamps at which it received the *Response* frames from each anchor.

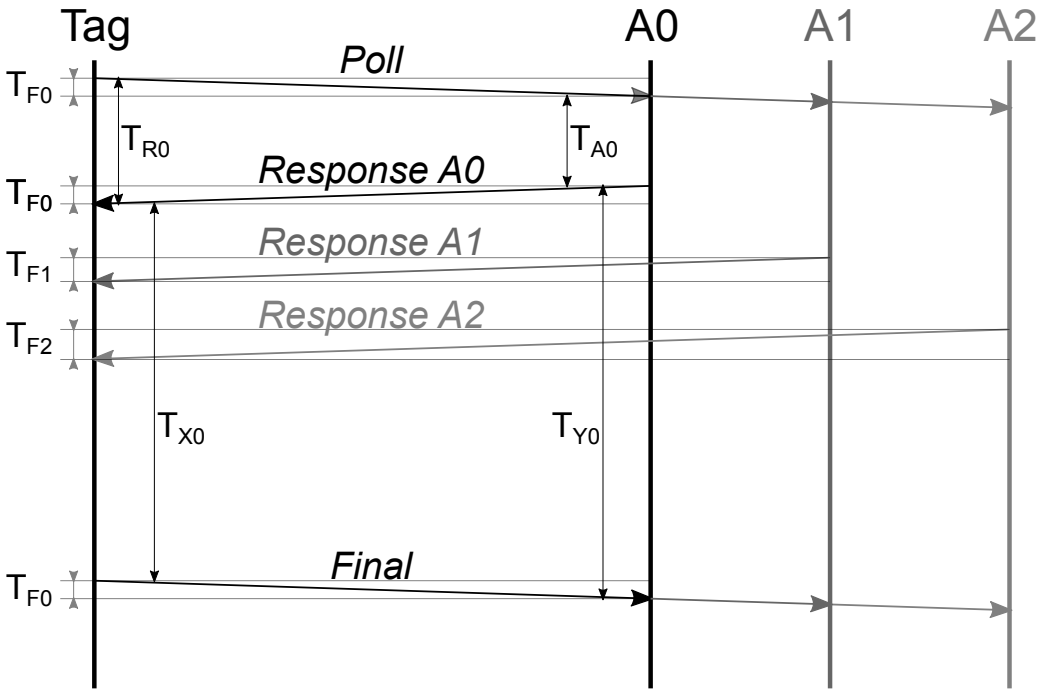


Fig. 3.2: Data exchange in a single slot

The time of flight (ToF) is determined from the delays during the exchange of the three messages: *Poll*, *Response* and *Final*. In the diagram it is denoted $T_{\mathrm{F}i}$, where i is the address of the anchor that the measured time refers to. For clarity, most of the labels for anchors other than A_0 have been left out. The calculations assume that the speeds of the oscillators in the individual devices (denoted α for the tag and β for the anchor) may differ from each other, but do not change during a single slot. The system of equations[(2)] for the time of flight to anchor A_0 is given in (3.1).

[(2)] Bitcraze Forums: *DWM1000 DS-TWR method calculate distance method?*, `https://forum.bitcraze.io/viewtopic.php?t=1944#p9959` (accessed 05.09.2018)

$$\begin{cases} \alpha \cdot T_{\mathrm{R0}} = 2 \cdot T_{\mathrm{F0}} + \beta \cdot T_{\mathrm{A0}} \\ \beta \cdot T_{\mathrm{Y0}} = 2 \cdot T_{\mathrm{F0}} + \alpha \cdot T_{\mathrm{X0}} \\ \dfrac{\alpha + \beta}{2} = 1 \end{cases} \tag{3.1}$$

The third equation follows from the assumption that the arithmetic mean of α and β cancels out the difference between the speeds of the devices' oscillators. Solving the system gives the formula for the time of flight (3.2) (also given in the documentation [8] and [9]).

$$T_{\mathrm{F0}} = \frac{T_{\mathrm{R0}} \cdot T_{\mathrm{Y0}} - T_{\mathrm{A0}} \cdot T_{\mathrm{X0}}}{T_{\mathrm{R0}} + T_{\mathrm{A0}} + T_{\mathrm{X0}} + T_{\mathrm{Y0}}} \tag{3.2}$$

To obtain the distance, the time is then multiplied by the speed of light in air, c = 299 702 547 m/s, and corrected for the antenna delays at the selected radio frequency. At a data rate of 6.8 Mbps, one slot lasts 10 ms. With 10 slots in a superframe, the standard position update rate is only 10 Hz. The system tested in this thesis has only one tag and three anchors, so the number of slots has been reduced to three: one for the tag and two for autopositioning. This raised the rate to about 33 Hz and cut the delay from 100 ms to 30 ms (the new duration of a superframe), which makes mixed reality applications run more smoothly. The **C** source code of the board firmware, needed for this modification, was provided by the manufacturer. The code was edited and compiled in the **Ac6 System Workbench for STM32** environment, available on the **OpenSTM32 Community** platform[(3)]. It is compiled with the **GNU ARM Toolchain** for the `arm-none-eabi` architecture. The **EVB1000** boards have a JTAG connector and SWD (Serial Wire Debug) pins, labelled `J4`, compatible with **ST-Link** programmers. This thesis used the latter, with the programmer shown in Figure 3.3. The connections used are listed in Table 3.1.

Each **EVB1000** board has a **USB** port for communicating with the system. The firmware uses a virtual UART-compatible port, whose driver is provided by the mi-

[(3)] OpenSTM32 Community: *System Workbench for STM32*, `http://www.openstm32.org/System+Workbench+for+STM32` (accessed 05.09.2018)

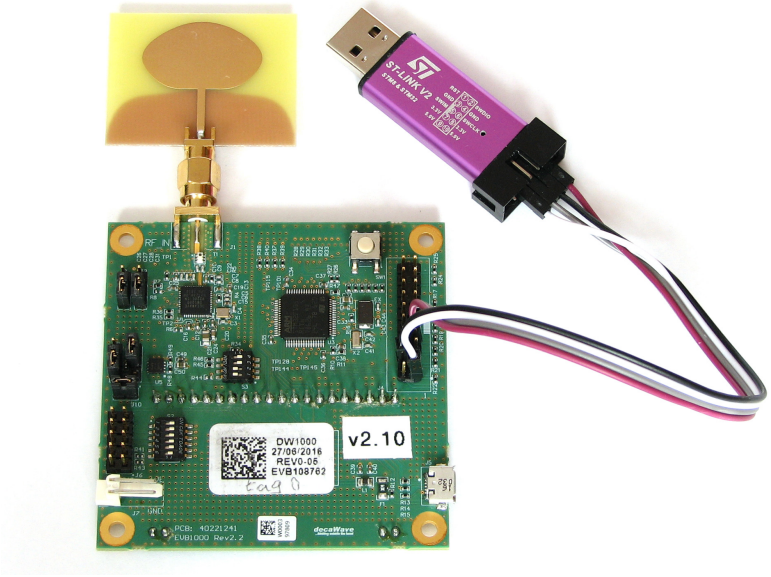


Fig. 3.3: ST-Link/V2 programmer used to program the EVB1000 boards

Pin	Name	Description
1	VDD3V3	Board power supply 3.3 V
4	GND	Power ground
7	JTMS-SWDAT	SWD data line (SWDIO)
9	JTCK-SWCLK	SWD clock line (SWCLK)

Tab. 3.1: Connections for programming the EVB1000 board

crocontroller manufacturer, **STMicroelectronics**. Because of the way the kit works, connecting to a tag only gives the distances between that tag and the anchors. To get data from all devices, one has to connect to the main anchor, with address `0x00`.

Once connected, the device starts sending the results of the distance measurements made between the devices of the **TREK1000** system that are configured with the same radio parameters. The format of a single data packet is as follows [9][10]:

```
MID MASK RANGE0 RANGE1 RANGE2 RANGE3 NRANGES RSEQ DEBUG aT:A
```

Each packet ends with the newline sequence `CRLF`. An example packet (for a measurement between the tag and the anchors) is shown below:

```
mc 0f 00000663 000005a3 00000512 000004cb 095f c1 00024c24 a0:0
```

It consists mainly of hexadecimal values in text form and contains the following information:

1) `MID` – packet identifier, one of the following values:
    - `mr` – uncorrected measurements between the tag and the anchors;
    - `mc` – as "`mr`", with corrections and antenna delays applied;
    - `ma` – measurement between anchors, i.e. autopositioning (with corrections applied).
2) `MASK` – bit mask showing which of `RANGE0` to `RANGE3` are valid;
3) `RANGE0..3` – for tag-to-anchor measurements (`mr` and `mc`), the distance (in mm) from the tag to the successive anchors, with addresses A0 to A3; for autopositioning (`ma`):
    - `RANGE0` – ignored;
    - `RANGE1` – distance from anchor A0 to anchor A1;
    - `RANGE2` – distance from anchor A0 to anchor A2;
    - `RANGE3` – distance from anchor A1 to anchor A2.
4) `NRANGES` – number of valid measurements made so far;
5) `RSEQ` – sequence number of the packet for the given `MID`;
6) `DEBUG` – for an "`ma`" packet, the measured antenna delay; otherwise, the time at which the last measurement was received;
7) `aT:A` – address (identifier) of the tag (`T`) and of the anchor (`A`) that the measurement refers to.

3.2. Server application

A server application has been developed to receive the measurements from the DecaWave TREK1000 system and pass them on to the mobile application. It is written in C++, using the cross-platform Qt framework. This means it can run on the most popular operating systems, and its graphical interface makes it intuitive, so that it

can be used by someone without detailed knowledge of how the system works. The Eigen library[4] is used for the coordinate transformations.

The application is shown in Figure 3.4. At the top of the window is a list of the available serial ports, from which the connected **EVB1000** board should be selected. The device name should contain the phrase "`STMicroelectronics Virtual COM Port`". On Windows, a driver has to be installed first[5]. Pressing the "Connect" button opens the serial port and starts communication.

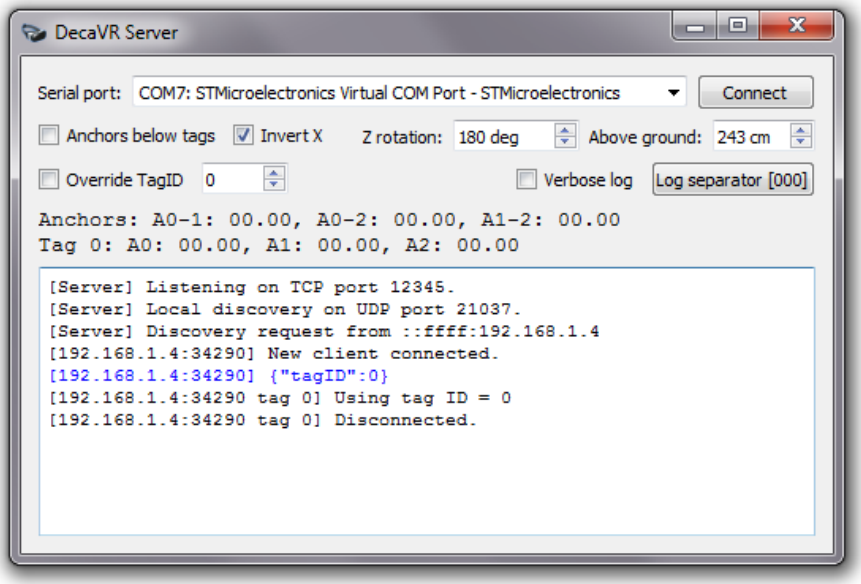


Fig. 3.4: The server application

Communication with the device is handled by the Qt class `QSerialPort`. The format of the received data is described in Section 3.1, on page 26. First, the autopositioning measurements between the anchors, received at 33 Hz, are collected. Thanks to this, the user does not have to enter the positions of the modules by hand. The data is then smoothed with a moving average filter with a window length of 50. This reduces the inaccuracy and interference caused by people moving within range of the system, and makes the position reading more stable. The positions of the anchors in the XY coordinate system are determined from three circle equations. To simplify the calculations, all anchors are assumed to be at the same height, which

[4] Eigen: `http://eigen.tuxfamily.org/index.php?title=Main_Page` (accessed 05.09.2018)
[5] STMicroelectronics: *STSW-STM32102 - STM32 Virtual COM Port Driver*, `http://www.st.com/en/development-tools/stsw-stm32102.html` (accessed 05.09.2018)

the user enters in the "Above ground" field. It is also assumed that anchor A_0 is at the origin of the coordinate system and A_1 on the positive X axis. If the placement of the receivers requires the X axis to be inverted, the "Invert X" option should be checked in the program, or A_0 and A_1 swapped. The position of anchor $A_2(x, y)$ can be determined from the circle equations (3.3). The distance from anchor A_i to A_j is denoted R_{ij}.

$$\begin{cases} A_0 : x^2 + y^2 = R_{02}^2 \\ A_1 : (x - R_{01})^2 + y^2 = R_{12}^2 \end{cases} \tag{3.3}$$

Solving them gives the formulas for the coordinates of the third anchor (3.4). The layout, with the distances and the intersections of the circles, is shown in Figure 3.5.

$$\begin{cases} x = \dfrac{R_{01}^2 + R_{02}^2 - R_{12}^2}{2R_{01}} \\ y = \pm\sqrt{R_{02}^2 - x^2} \end{cases} \tag{3.4}$$

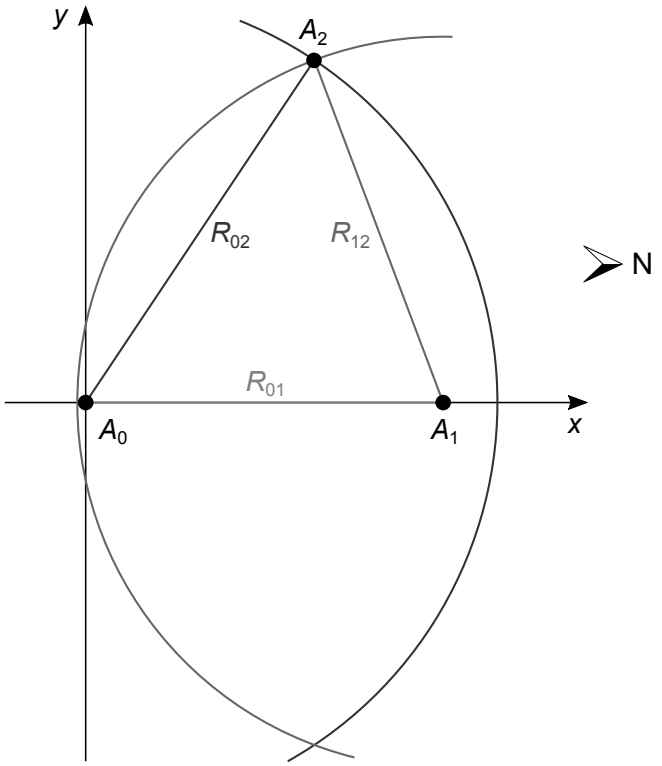


Fig. 3.5: Geometry of anchor autopositioning

Of the two possible solutions, the one with a positive y coordinate is chosen, so

the point lies in the first or second quadrant of the coordinate system. This allows the position to be translated correctly into the virtual world. Next, the program processes the measurements between the tag and the anchors. The distances are converted into coordinates in the virtual world by trilateration. The anchors are assumed to be at the same height, which simplifies the initial equations (3.5) without loss of accuracy. If the heights differ, the result can be corrected by rotating the coordinate system.

$$\begin{cases} A_0 : x^2 + y^2 + z^2 = R_0^2 \\ A_1 : (x - A_{1x})^2 + y^2 + z^2 = R_1^2 \\ A_2 : (x - A_{2x})^2 + (y - A_{2y})^2 + z^2 = R_2^2 \end{cases} \tag{3.5}$$

Solving the system of equations (3.5) gives the coordinates of the tag, $T(x, y, z)$ (3.6). The solution in the plane defined by the anchors is shown graphically in Figure 3.6.

$$\begin{cases} x = \dfrac{A_{1x}^2 + R_0^2 - R_1^2}{2A_{1x}} \\ y = \dfrac{A_{2x}^2 + A_{2y}^2 + R_0^2 - R_2^2}{2A_{2y}} - \dfrac{A_{2x}}{A_{2y}} x \\ z = \pm\sqrt{R_0^2 - x^2 - y^2} \end{cases} \tag{3.6}$$

Which solution is chosen for the z coordinate depends on the "Anchors below tags" setting in the server application. When this option is checked, the positive solution is chosen, which places the tag above the anchors; otherwise, the negative one. The coordinate system is then transformed so that the centroid of the triangle formed by the anchors is at the origin. As a result, the central parts of the virtual and of the real room are in approximately the same place. The height of the anchors above the ground should be entered in the "Above ground" field; the calculated position of the tag, and thus the height in the virtual world, is shifted by this value. The coordinate system is then rotated to match the readings of the phone's compass. The vector $\overrightarrow{A_0A_1}$, lying on the X axis, is assumed to point north. If it does not, the angle in degrees should be entered in the "Z rotation" field. For example, a value of

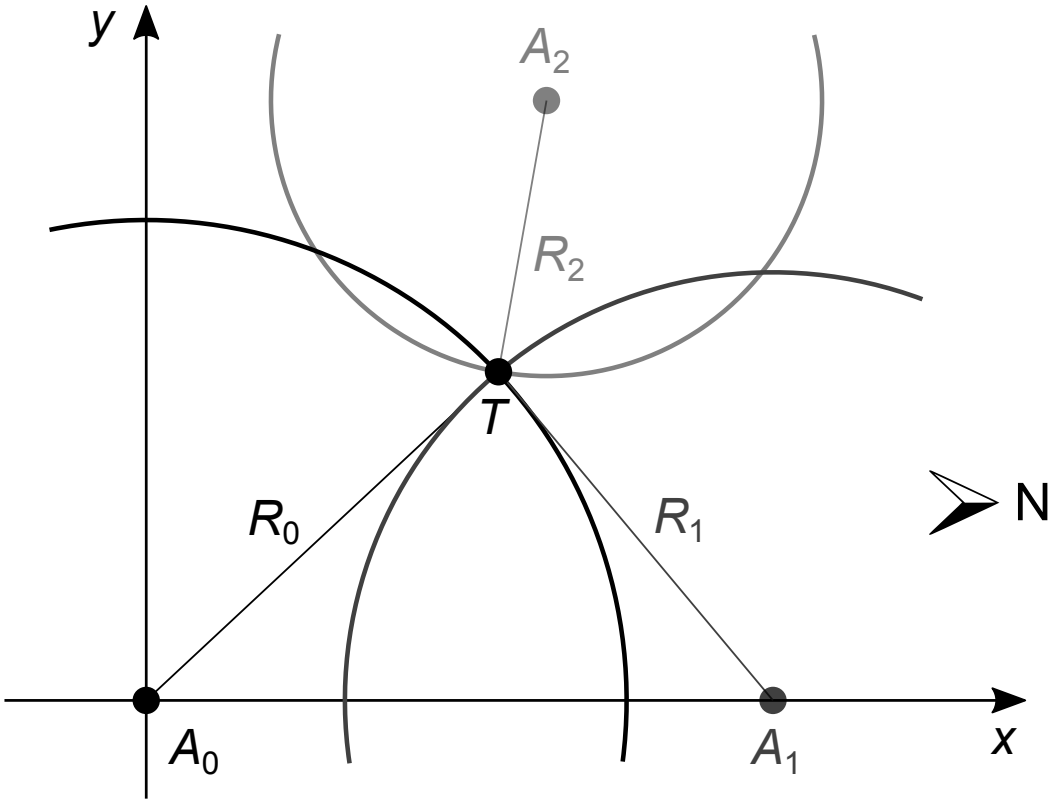


Fig. 3.6: Graphical representation of tag trilateration in the plane of the anchors

"90" means that the vector $\overrightarrow{A_0A_1}$ points east.

The current measurements are displayed in the middle panel of the application: both the autopositioning distances between the anchors, with their calculated coordinates, and the position of the tag with identifier `0x00`. All collected data can be written to a text log file named "`log.txt`" by checking "Verbose log". The button next to "Log separator [000]" inserts separators into the collected data, which makes later analysis easier. An example log with a separator is shown in Listing 3.1.

The location is sent from the server application to the mobile one wirelessly, over a local Wi-Fi network. For this, the computer and the smartphone are connected to a common access point on the same subnet. Right after the server starts, it begins listening on fixed ports over the UDP and TCP protocols. UDP is used to discover the server's address: the client sends a *broadcast* datagram with the content "`DecaVR Discovery`" to all devices on the subnet, on port 21037. On receiving such a message, the server replies in the same way with "`DecaVR Server:12345`",

```
17:05:59.743:[Status] Anchor coords: A1:2.76,0.00, A2:1.29,-4.05
17:05:59.744:[Status] Anchors: 01: 2.80, 02: 4.28, 12: 4.30 [70]
17:05:59.758:[Status] Tag 0: A0: 2.79, A1: 2.90, A2: 2.86 [157]
17:05:59.758:[Status] Location:  0.08,  0.43,  0.70
17:05:59.776:[Status] Anchor coords: A1:2.76,0.00, A2:1.29,-4.05
17:05:59.776:[Status] Anchors: 01: 2.77, 02: 4.25, 12: 4.34 [71]
17:05:59.786:[Status] Tag 0: A0: 2.78, A1: 2.89, A2: 2.88 [158]
17:05:59.786:[Status] Location:  0.08,  0.41,  0.69
17:06:00.955:[MAIN] -------- LOG SEPARATOR 001 --------
17:06:58.122:[Status] Anchor coords: A1:2.76,0.00, A2:1.29,-4.05
17:06:58.122:[Status] Anchors: 01: 2.75, 02: 4.23, 12: 4.31 [224]
17:06:58.135:[Status] Tag 0: A0: 2.85, A1: 2.95, A2: 2.89 [55]
17:06:58.136:[Status] Location:  0.08,  0.44,  0.62
17:06:58.151:[Status] Anchor coords: A1:2.76,0.00, A2:1.29,-4.05
17:06:58.152:[Status] Anchors: 01: 2.74, 02: 4.28, 12: 4.31 [225]
17:06:58.166:[Status] Tag 0: A0: 2.82, A1: 2.94, A2: 2.87 [56]
17:06:58.166:[Status] Location:  0.10,  0.45,  0.65
```

Listing 3.1: Verbose log with a separator

where `12345` is the number of the TCP port on which it listens for incoming connections. This lets the client obtain the server's parameters and connect to it. The port number may vary, depending on which ports are free and how the operating system assigns them. The network must be configured to allow broadcast frames, so that the server can be discovered.

The actual communication uses the more reliable TCP protocol. The connection is handled by the Qt class `QTcpSocket`. To minimize latency, the `TCP_NODELAY` option, which disables Nagle's algorithm, is enabled through the socket option `QAbstractSocket::LowDelayOption`. As a result, small chunks of data are not buffered but sent immediately, whatever their size. Messages between the devices are sent as text: JSON objects terminated by the newline sequence `CRLF`. This makes the data easy to read both for humans and for programs. Most technologies have tools for parsing and encoding this format, including the Qt framework, with its `QJsonDocument` and `QJsonObject` classes. After the client connects, the server waits for the identifier of the tag whose position is to be calculated and sent. It is given in the "`tagID`" parameter, of *integer* type. For example, after receiving the following request, the server starts sending the location of the tag with address `0x00`:

```
{"tagID": 0}
```

Depending on the identifier given, the server either connects to a physical device, as described at the beginning of this section (from page 28), or sends previously generated data. To make the system easier to test, a position simulator that moves between fixed points was written for this thesis (the `SimulatedLocationProvider` class). It is available under the `tagID` value `9`. Previously collected positions can also be replayed from logs saved in a text file: the application looks in the current directory for a log with the given identifier in its name. For example, for the file "`log42.txt`", the `tagID` `42` should be given. This functionality is provided by the `LoggedLocationProvider` class and works for identifiers greater than or equal to `10`. The values from `0` to `7` correspond directly to the physical address of a tag, whose measurements are then read through the serial port. These identifiers are handled by the `SerialLocationProvider` class.

If a packet received by the server application has an invalid format, the server replies with error information, also in JSON format. The message has the following form:

`{"error": "`*error code*`", "message": "`*detailed error description*`"}`

The "`message`" value describes the error in a way the user can understand. The "`error`" parameter contains one of the text values listed below and is meant for machine processing:

- `MALFORMED_MESSAGE` – the message is not valid JSON, or the "`tagID`" parameter is not an *integer*;
- `MISSING_PARAMETER` – the message has no "`tagID`" parameter;
- `UNKNOWN_TAG` – the requested tag identifier does not exist.

If the received packet is processed successfully, the server starts sending positions for the selected tag (physical or virtual). The parameters in each packet are listed in Table 3.2.

The "`timestamp`" value is used to tell successive measurements apart and may be the time in milliseconds since the start of the measurements. After each write to the network socket, the data is sent immediately by calling the `flush()` method.

Parameter	Type	Description
`timestamp`	*integer*	timestamp of the position
`x`	*integer*	x coordinate in cm
`y`	*integer*	y coordinate in cm
`z`	*integer*	z coordinate in cm

Tab. 3.2: Parameters of the position packets sent by the server

This removes the delay caused by Qt's internal buffering. The data sent over the network between the devices is also written to the text log, to help find any communication errors. The received tag identifier `tagID` can be overridden on the server side by checking "Override TagID" and setting the desired value in the number field next to it. This makes it possible to test previously collected measurements without changing the client application.

3.3. Mobile application

The mobile application was developed for the Android operating system, in the Unity 2017 environment. The Google VR SDK framework (formerly known as Google Cardboard)[(6)] is used to generate a stereoscopic image. This lets the scene be viewed in three dimensions, with depth perception. The screen is split into two halves, so that once the phone is placed in a virtual reality (VR) headset, each eye receives a separate image. In addition, a special filter is applied to correct the lens distortion at the edges of the field of view.

This master's thesis used the **VR BOX 2.0** virtual reality headset, shown in Figure 3.7. It comes with a wireless Bluetooth remote, shown in Figure 3.8, which was used to interact with the mobile application.

The scene in the application is based on the examples supplied with Google VR SDK. It consists of a virtual room measuring 5 × 5 × 2.5 m. A square grid with 1 m cells is placed on the floor, which makes it easier to keep one's bearings while moving. An additional interactive element is a cube that emits sound. It can be moved by pointing at it and pressing the side button on the remote. For designing

[(6)] Google VR for everyone: `https://developers.google.com/vr/` (accessed 05.09.2018)



Fig. 3.7: The VR BOX 2.0 headset used in this thesis

and testing the system, a text field was added that shows the smoothness of the simulation in frames per second (FPS), the orientation from the phone's sensor and the position obtained from the TREK1000 system. A bird's-eye view of the scene in the editor is shown in Figure 3.9.

By default, the Google VR SDK framework does not support devices without a built-in gyroscope. For this reason, an add-on (plugin) was developed that reads the orientation from the *rotation vector sensor* available on Android devices. Its readings are usually based on a combination of the accelerometer, the compass and – if the hardware has one – the gyroscope [14]. The add-on is a library compiled into an Android Archive (AAR) [16][17]. It consists of the `OrientationProvider` class, which implements the `SensorEventListener` interface for receiving sensor readings. The selected sensor type is `TYPE_ROTATION_VECTOR`, with the `SENSOR_DELAY_GAME` update period of at least 20 ms, i.e. a rate of 50 Hz, which should be sufficient for virtual reality applications. The sensor measurements are given in a coordinate system that depends on the device's default orientation [15]. For smartphones this is usually portrait, so the axes are arranged as in Figure 3.10. The



Fig. 3.8: Bluetooth remote from the VR BOX 2.0 set

rotation vector sensor's readings are a unit quaternion representing the rotation from the reference position, which is defined as follows [14]:

- the x axis is tangent to the ground plane and points east (E);
- the y axis is tangent to the ground plane and points north (N);
- the z axis is perpendicular to the ground and points towards the sky.

The first four values received from the sensor S are given by equations (3.7), where φ is the rotation angle and x, y and z define the vector around which the rotation is made.

$$\begin{cases} S_0 = S_x = x \cdot \sin\left(\frac{\varphi}{2}\right) \\ S_1 = S_y = y \cdot \sin\left(\frac{\varphi}{2}\right) \\ S_2 = S_z = z \cdot \sin\left(\frac{\varphi}{2}\right) \\ S_3 = S_w = \cos\left(\frac{\varphi}{2}\right) \end{cases} \tag{3.7}$$

Some devices provide an additional value giving the orientation error caused by compass miscalibration. Accuracy can be improved by rotating the phone about

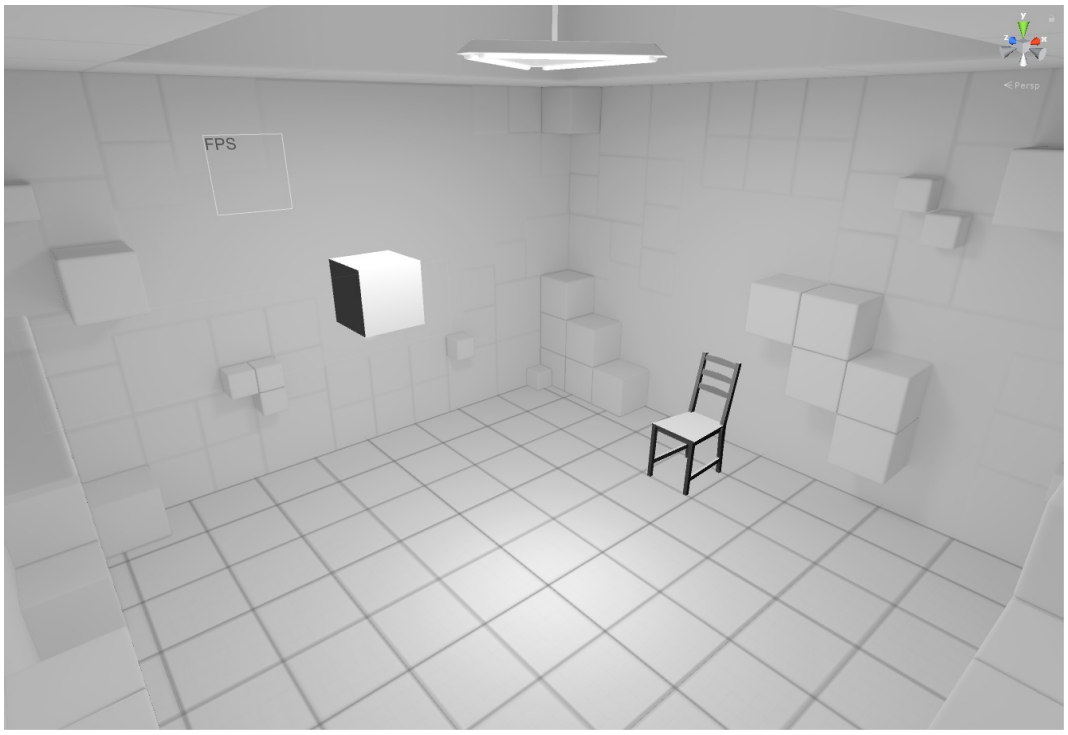


Fig. 3.9: The scene in the Unity editor

different axes, or by moving it a few times in a figure-eight pattern.[7] The measurements are read from the plugin, and the calculations made, in the `CameraRotation` script. The plugin class is instantiated and used through the reflection mechanism available in Unity. The Android sensor coordinate system is right-handed, whereas Unity's is left-handed. The received quaternion S therefore has to be transformed by swapping its components as shown in (3.8), which gives the quaternion $Q(x, y, z, w)$.

$$\begin{cases} Q_x = -S_y \\ Q_y = S_x \\ Q_z = -S_z \\ Q_w = -S_w \end{cases} \tag{3.8}$$

To give the best possible experience with the application, the parameters of the **VR BOX 2.0** headset were calibrated. Initial values were taken from ready-made

[7] Google Maps Help: *Find and improve your location's accuracy – Android*, `https://support.google.com/maps/answer/2839911?co=GENIE.Platform%3DAndroid&` (accessed 05.09.2018)

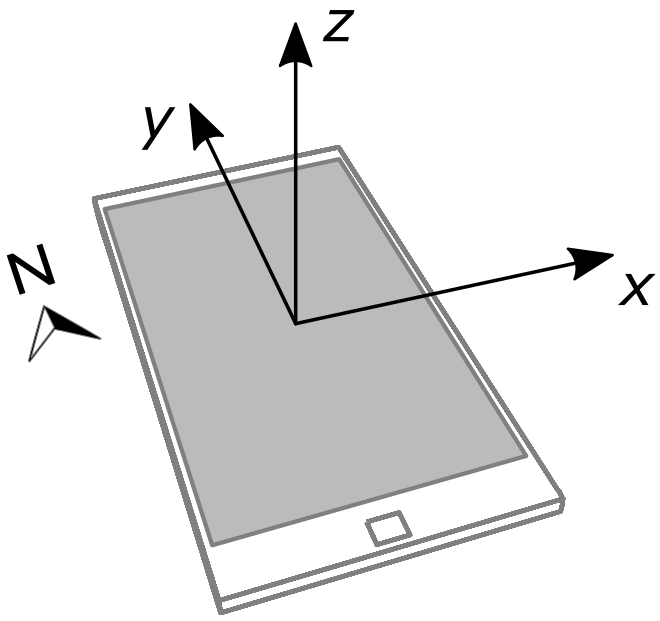


Fig. 3.10: Smartphone sensor coordinate system in Android

configurations available online.[8] Then, with the online Google Cardboard Viewer Profile Generator[9] the lens spacing, the distortion coefficients and the viewing angles were adjusted to the headset used. This produces the QR code shown in Figure 3.11, which can be scanned in the Google Cardboard configuration app; alternatively, a `current_device_params` file can be generated and placed in the phone's `/sdcard/Cardboard/` directory. The application running on the phone is shown in Figure 3.12.

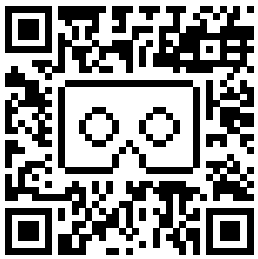


Fig. 3.11: QR code with the configuration of the VR BOX 2.0 headset

Communication with the server application described in Section 3.2 is imple-

[8] Hypergrid Business: *VR Headset QR Codes*, `https://www.hypergridbusiness.com/faq/vr-headset-qr-codes/` (accessed 05.09.2018)
[9] Google: *Cardboard viewer profile generator*, `https://wwgc.firebaseapp.com/` (accessed 05.09.2018)

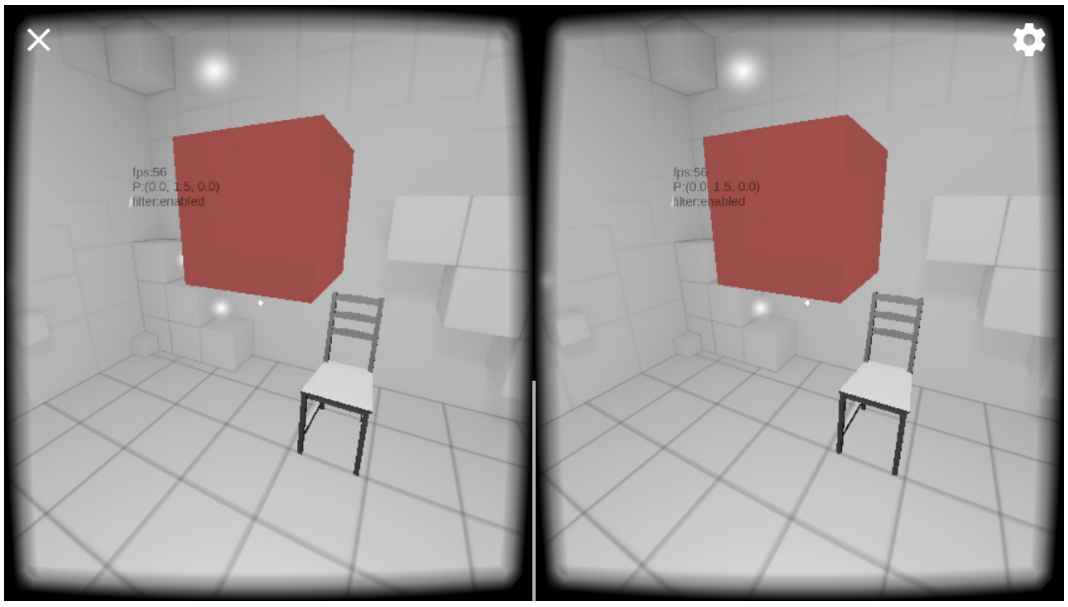


Fig. 3.12: Screenshot of the mobile application on the phone

mented in the `NetworkClient` script. It uses the `UdpClient` and `TcpClient` classes, available in .NET, which make working with network sockets convenient. JSON objects are handled with the `JsonUtility` module provided by Unity. When a position is received, its coordinates are smoothed with an alpha-beta (α-β) filter. It is a simplified derivative of the Kalman filter that needs much less computation and gives satisfactory results when filtering objects that move with small accelerations [18]. Its name refers to the parameters α and β, which determine how much each new measurement affects the filtered value. The filter equations are given in (3.9). The function $m(t)$ is the measurement taken at a given moment; ΔT is the time since the previous measurement; $x(t)$ is the predicted value – the location of the tracked object; and $v(t)$ is its derivative – here, the velocity.

$$\begin{cases} p(t) = x(t-1) + v(t-1) \cdot \Delta T \\ r = m(t) - p(t) \\ x(t) = p(t) + \alpha \cdot r \\ v(t) = v(t-1) + \dfrac{\beta \cdot r}{\Delta T} \end{cases} \tag{3.9}$$

The filter coefficients were chosen experimentally: $\alpha = 0.065$ and $\beta = 0.002$.

Increasing the first one improves the response to rapid changes of position, but at the same time makes the filter less resistant to noise. Too high a value of β introduces unwanted oscillations around the mean value.

The tag, an EVB1000 board, was attached to the VR headset so that it is in direct line of sight of the three anchors on the ceiling at all times. In practice, this means it is placed on the user's head. The height difference between the antenna and the eyes was taken to be about 20 cm, and the mobile application corrects the camera position by this amount. The device is powered through a USB cable from a battery placed at the bottom of the headset. A photo of a user wearing the headset with the tag is shown in Figure 3.13.



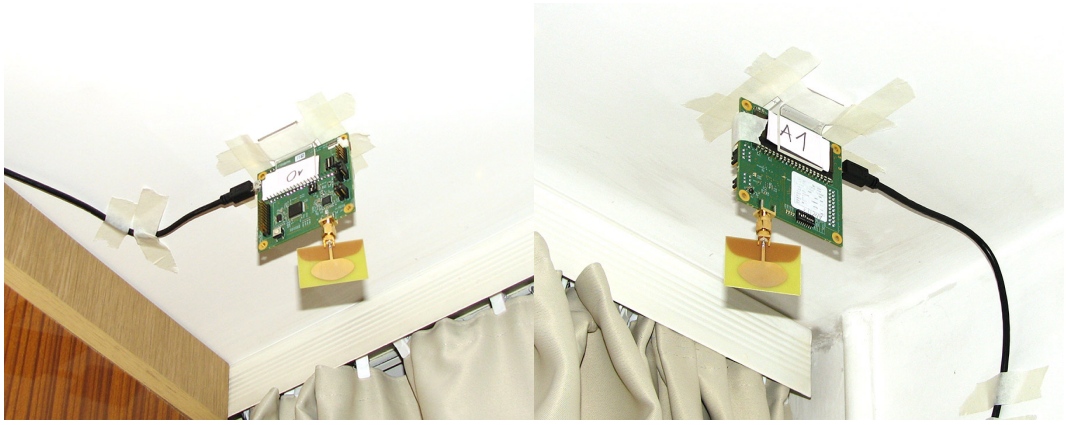
Fig. 3.13: User wearing the headset with the EVB1000 tag

Chapter 4

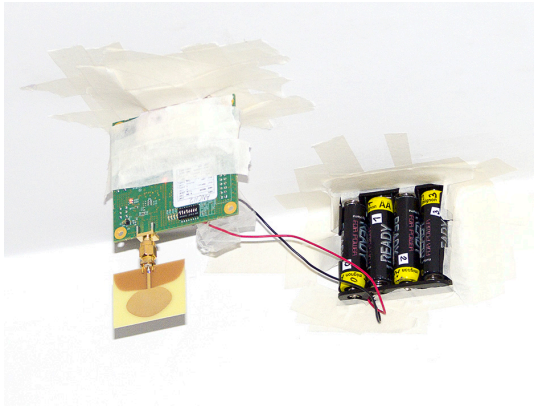
Performance tests

To determine the accuracy of the system, numerous measurements were made to establish whether it is suitable for the application intended at the outset. The tests were divided into static tests, which show the deviations from the exact value while the device stays in one place; dynamic tests, which show how well the system tracks a moving object; and subjective tests, in which the user assesses the quality of operation. The anchors were attached to the ceiling at a height of (253.0 ± 0.5) cm and at least 15 cm from the nearest wall or obstacle, as recommended by the manufacturer [7]. The layout was designed to form an isosceles triangle. The antennas of the anchors were pointed towards the center of the room. Their positions are shown as square symbols in Figure 4.2, and a photo of each of them after mounting is shown in Figure 4.1. The calculations and charts were made in Python, with the NumPy and Matplotlib libraries. The main anchor, A_0, was powered by connecting it to a Lenovo ThinkPad X201 laptop, which at the same time communicated with the whole system over the serial link and read the measurements. Anchor A_1 was connected to an EnerGenie EG-UC2A-01 USB power supply/charger[(1)] with a maximum output current of 2.1 A. Anchor A_2 was powered by four Hama Ready For Power Ni-MH AA (HR03) rechargeable cells connected in series, each with a nominal voltage of 1.2 V and a capacity of 2100 mAh.

[(1)] EnerGenie: *Universal USB charger, 2.1 A*, `https://energenie.com/item.aspx?id=8618` (accessed 05.09.2018)



(a) Anchor A_0 (b) Anchor A_1



(c) Anchor A_2

Fig. 4.1: Anchors mounted for the tests

4.1. Autopositioning

The first stage of the tests checked whether the coordinates of the anchors are determined correctly by the automatic procedure. This uses the communication between the stationary modules of the DecaWave system, described in Section 3.2. The stability of the determined coordinates and their differences from the expected values were examined. The charts in Figure 4.3 are histograms of the 36,000 distance measurements made between the anchors during the tests. The mean values and standard deviations σ of the samples are given in Table 4.1, together with the expected values, which were measured by hand.

Then, using formulas (3.3) and (3.4), the basic coordinates of the anchors (before the transformations) were determined and compared with the expected ones. These values are listed in Table 4.2.

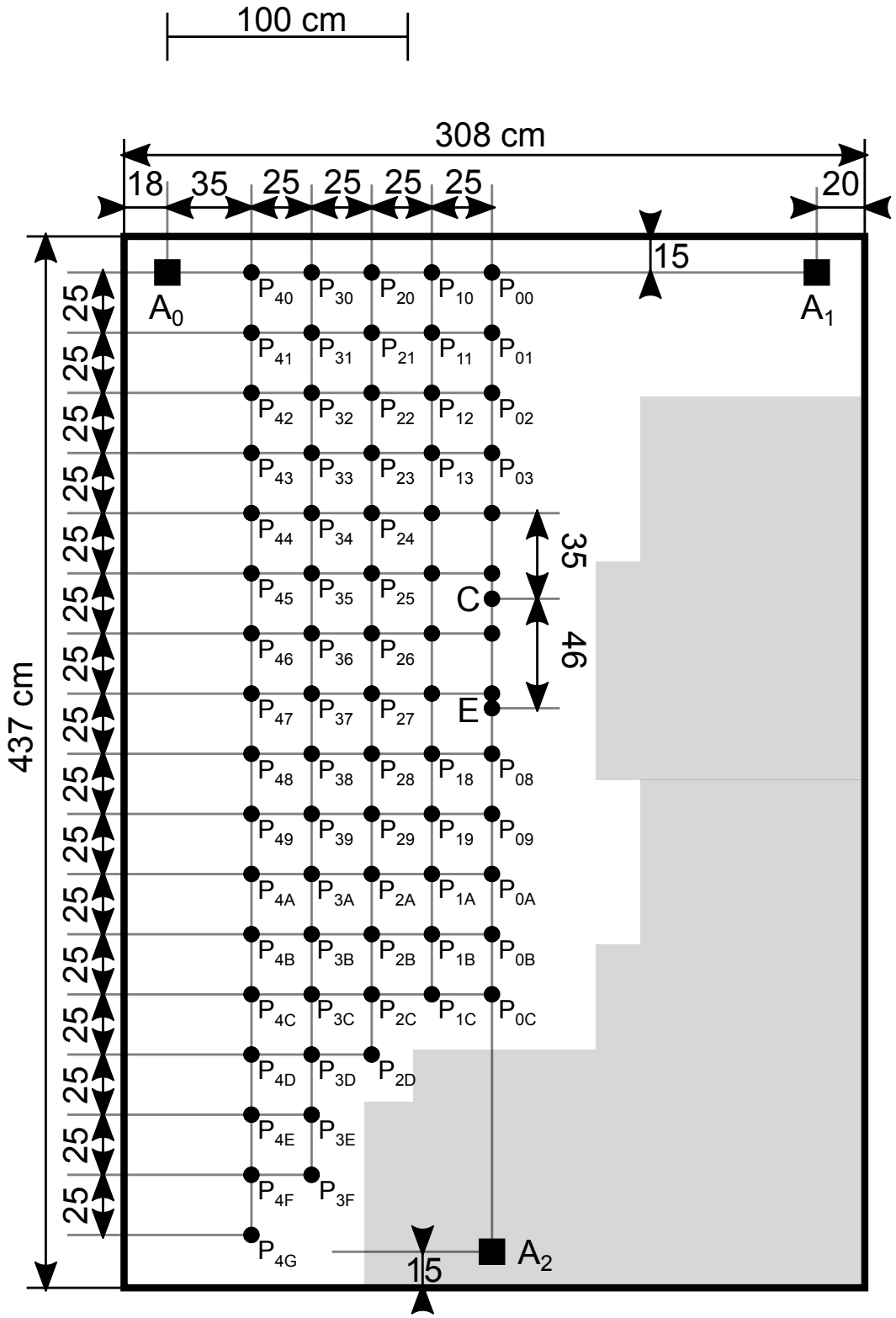
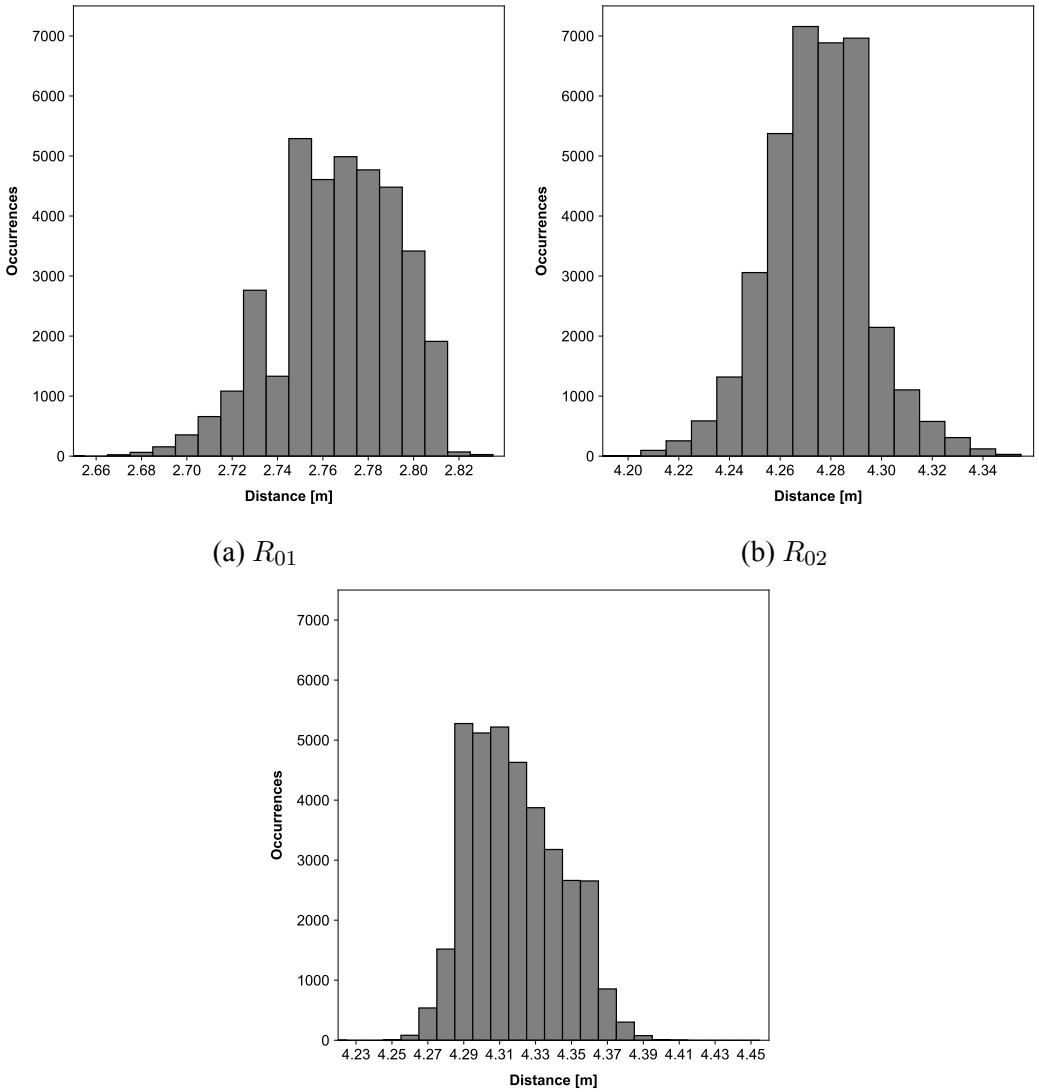


Fig. 4.2: Layout of the anchors and the static measurement points in the room (dimensions in cm)



(a) R_{01} (b) R_{02}

(c) R_{12}

Fig. 4.3: Histograms of the autopositioning distances

4.2. Static measurements

For the static tests, several dozen points 25 cm apart were laid out in the test room. Their layout, the positions of the system's transmitters and the dimensions of the room are shown in Figure 4.2. The TREK1000 transceiver working as a tag was attached to a tripod together with a 2200 mAh power bank that supplied it through the USB connector, as shown in Figure 4.4. In addition, two special points, C and E, were marked. The first is the centroid of the triangle $A_0A_1A_2$ formed by the anchors, which is also the origin of the virtual scene's coordinate system (as described in

Distance [m]	Expected	Mean	Difference	σ
R_{01}	2.700	2.767	0.067	0.026
R_{02}	4.288	4.275	-0.013	0.020
R_{12}	4.288	4.319	0.031	0.025

Tab. 4.1: Autopositioning distance statistics

Coordinates [m]	Expected	Mean	Difference
A_0	(0.000, 0.000)	(0.000, 0.000)	(0.000, 0.000)
A_1	(0.000, 2.700)	(0.000, 2.767)	(0.000, 0.067)
A_2	(1.350, 4.070)	(1.315, 4.068)	(-0.035, -0.002)

Tab. 4.2: Determined autopositioning coordinates

Section 3.2). It was chosen for the tests because the user will probably spend most of the time around it. Point E is at the same distance from all anchors, (226 ± 1) cm in the horizontal plane, so it is the center of the circle circumscribed about this triangle.

These measurements make it possible to compare the accuracy of each anchor and to determine how the orientation of the tag's antenna affects the readings of the devices. The position of point E was determined with formula (4.1), assuming that anchor A_0 is at the origin of the coordinate system. Gray shading marks obstacles that made it impossible to place the tripod stably at those spots, and thus to take measurements there with the required accuracy. Points P_{00} to P_{0C} lie on a line equidistant from anchors A_0 and A_1, and all the following ones, from P_{10} to P_{4G}, closer to where A_0 is mounted.

$$\begin{cases} D = 2(A_{1x}A_{2y} - A_{1y}A_{2x}) \\ E_x = \dfrac{1}{D}\left[A_{2y}(A_{1x}^2 + A_{1y}^2) - A_{1y}(A_{2x}^2 + A_{2y}^2)\right] \\ E_y = \dfrac{1}{D}\left[A_{1x}(A_{2x}^2 + A_{2y}^2) - A_{2x}(A_{1x}^2 + A_{1y}^2)\right] \end{cases} \tag{4.1}$$

The point where the antenna joins the EVB1000 board was taken as the reference point for the position measurements. For the tag on the tripod, this point was at a height of (155.0 ± 0.5) cm, which is (88.0 ± 0.5) cm from the plane formed by the antennas of the anchors at the ceiling. The exact position of the tripod was indicated by a weight hanging from it on an inextensible thread. The setup at one of the points



Fig. 4.4: Tripod with the tag during the static measurements

is shown in Figure 4.5.

At each point, 200 measurements of the distances from the anchors to the tag were collected for two antenna orientations: parallel to the longer (|) and to the shorter (–) wall of the room. This was needed because the system's readings differ depending on the orientation of the antenna planes. The mean values $\overline{x}$ and standard deviations σ were then calculated, and the results compared with the expected values m. The data is presented as the map in Figure 4.6. A radial chart divided into 6 parts is drawn at each point. The filled part shows the offset of the distance reading between the tag and the given anchor from the expected value at that point, and the black outline shows the standard deviation σ of the measurements.

At points that deviated significantly from the expected values, the measurements were repeated to be sure. These were mainly spots near the walls and close to the



Fig. 4.5: Weight indicating the exact position of the tripod

anchors. The results were similar, which rules out errors that could have come from a badly performed measurement, for example an inaccurately placed tripod. It was also checked whether the accuracy of position determination changes noticeably with the distance between the tag and the anchor. The chart in Figure 4.7 shows the differences from the expected distances, with the standard deviations, collected during the static measurements.

The orientation of the antenna around the vertical axis had a noticeable effect on the measured distance between the devices. To examine this relationship more closely, samples were collected at point E while the tag was rotated by hand on the tripod. After initial smoothing, the measurements were plotted against the angle on the radial chart in Figure 4.8. The data was normalized so that the parallel orientation of the tag's and the anchor's antennas is at 0°. The circle marking the expected distance from point E is drawn in red.

At the same point, a test was carried out to check how a person blocking the line of sight between the transmitters (LOS) affects the measurements. A person walked slowly around the tripod with the tag, at a distance of about one meter. Figure 4.9 shows the distances to all three anchors measured during this test. For readability, the results were filtered with a moving average 30 samples wide, which corresponds

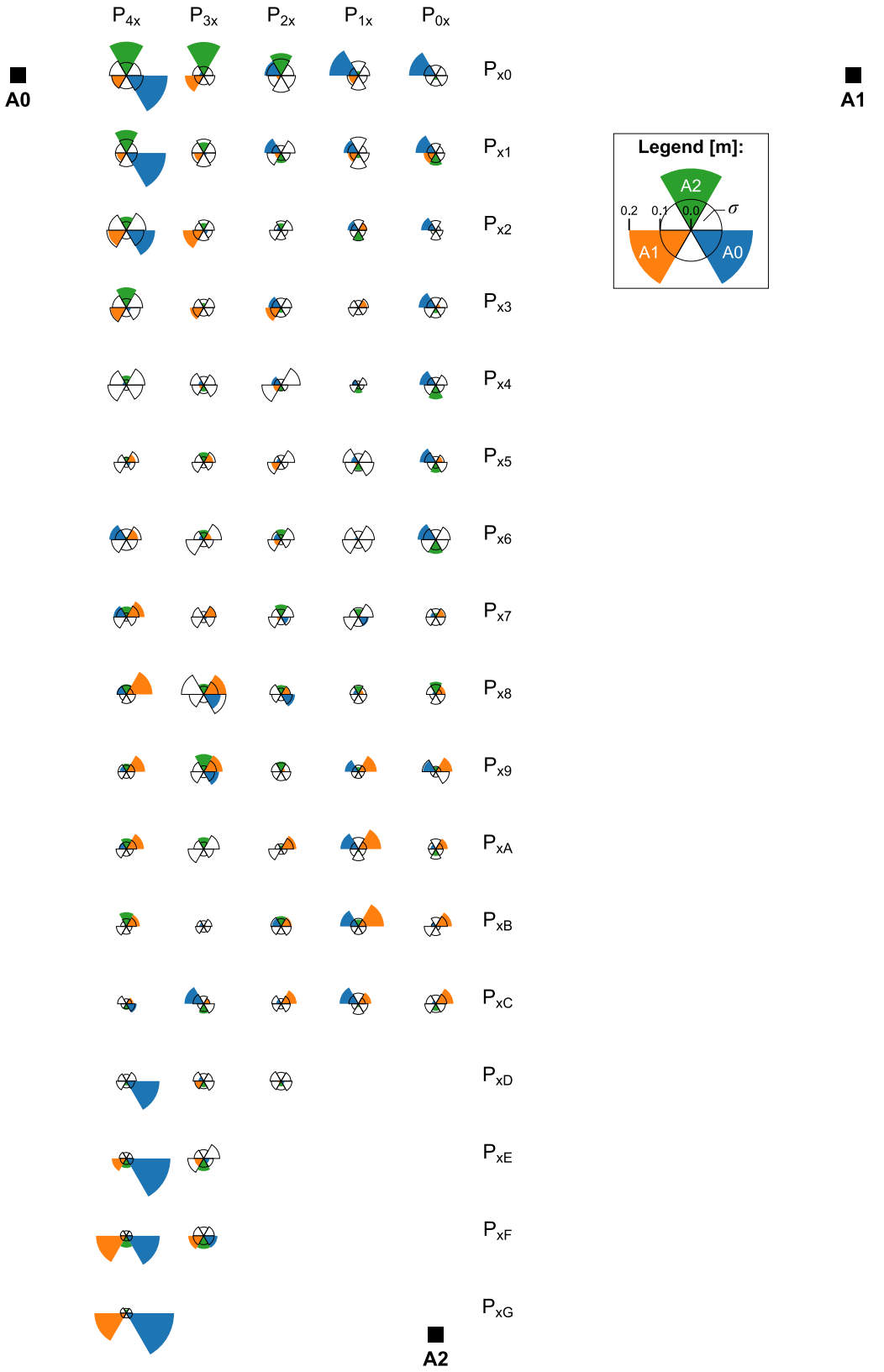


Fig. 4.6: Map of the reading accuracy at the static points

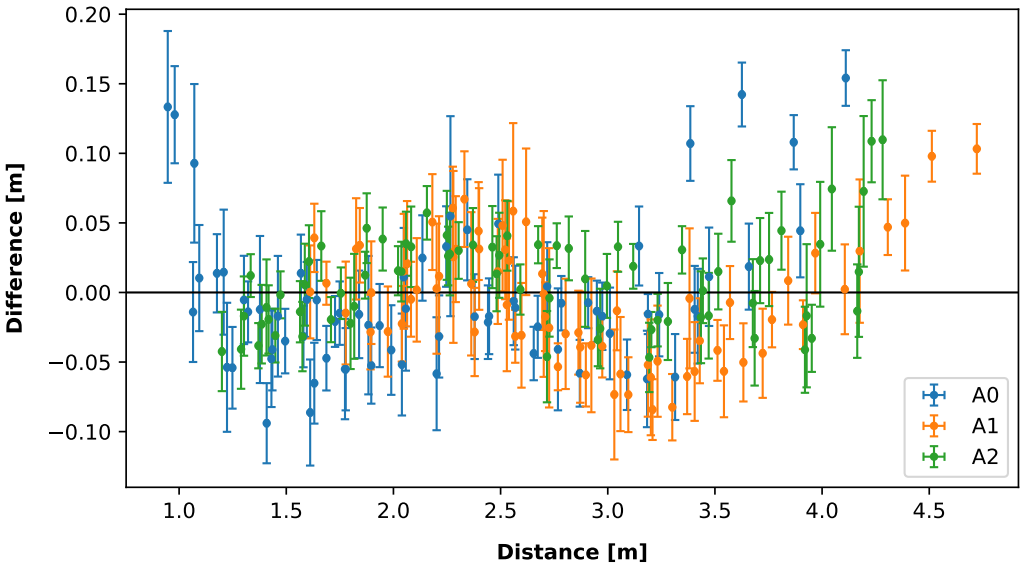


Fig. 4.7: Position accuracy as a function of the distance from the anchor

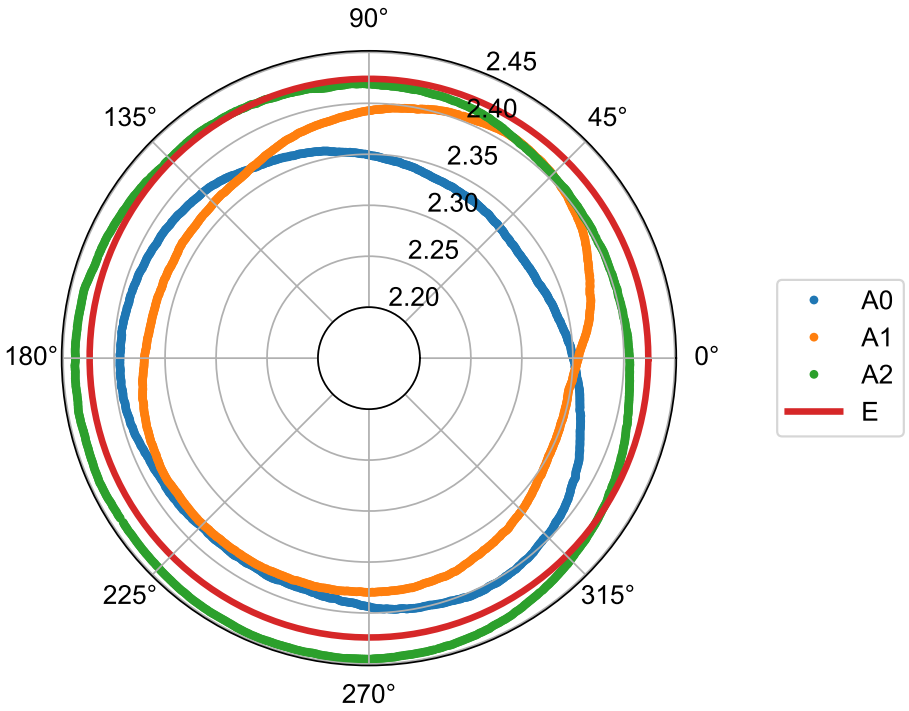


Fig. 4.8: Distance readings at point E depending on the antenna orientation

to one second of data collection. Deviations $\boldsymbol{d}_{\mathbf{avg}}$ from the mean value (2.390 m) can be seen at the moments when the person approached and blocked successive transmitters. The extreme readings $\boldsymbol{x}$ are listed in Table 4.3, together with their differences from the theoretical distance between point E and the anchors, 2.424 m.

As can be seen, the system's average readings were lower than this value; however, getting repeatable readings at the same points matters more than their exact values.

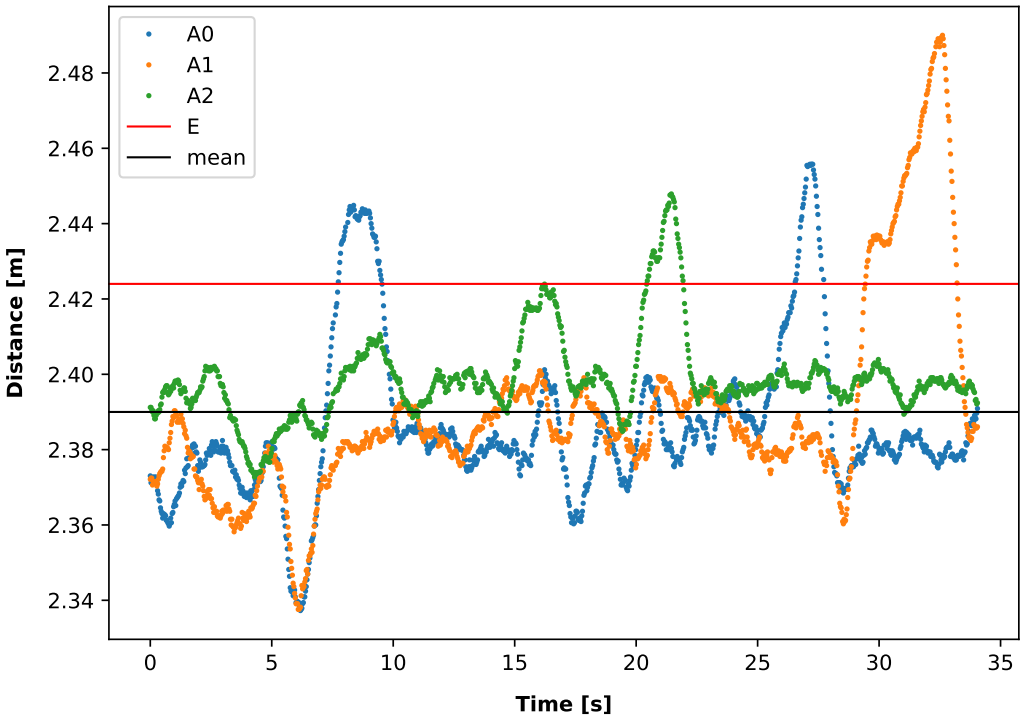


Fig. 4.9: Effect of a person blocking the transmitters on the distance readings

t [s]	A_i	x [m]	d_{avg} [m]	d_{E} [m]
6.2	A_0, A_1	2.337	-0.053	-0.087
8.4	A_0	2.445	0.055	0.021
16.2	A_2	2.423	0.033	-0.001
17.5	A_0	2.360	-0.030	-0.064
21.5	A_2	2.448	0.058	0.024
27.1	A_0	2.455	0.065	0.031
28.6	A_1	2.360	-0.030	-0.064
32.6	A_1	2.490	0.100	0.066

Tab. 4.3: Deviations of the distance readings after blocking the transmitters

4.3. Dynamic measurements

To determine the accuracy of the system during movement, a pendulum test was carried out. The tag, with its battery, was hung from the ceiling of the test room

on a long inextensible thread. The displacement was measured with a tape measure stretched parallel to the direction of the pendulum's swing. Figure 4.10 shows the pendulum at rest, with the antenna 100 cm from the start of the tape. To make the readings easier, a camera pointed at the moving tag was used. The pendulum swung along the Y axis, and its movement along the X axis was considered negligible. The thread was $l = (137.0 \pm 0.5)$ cm long, and the coordinates of the pendulum's suspension point were determined as $Q(1.26, 1.80, 0.4)$. The equations describing the oscillation are given in (4.2). The pendulum was assumed to oscillate harmonically, with amplitude A, frequency f, exponential damping coefficient α and offset t_0 relative to the start of the oscillations.

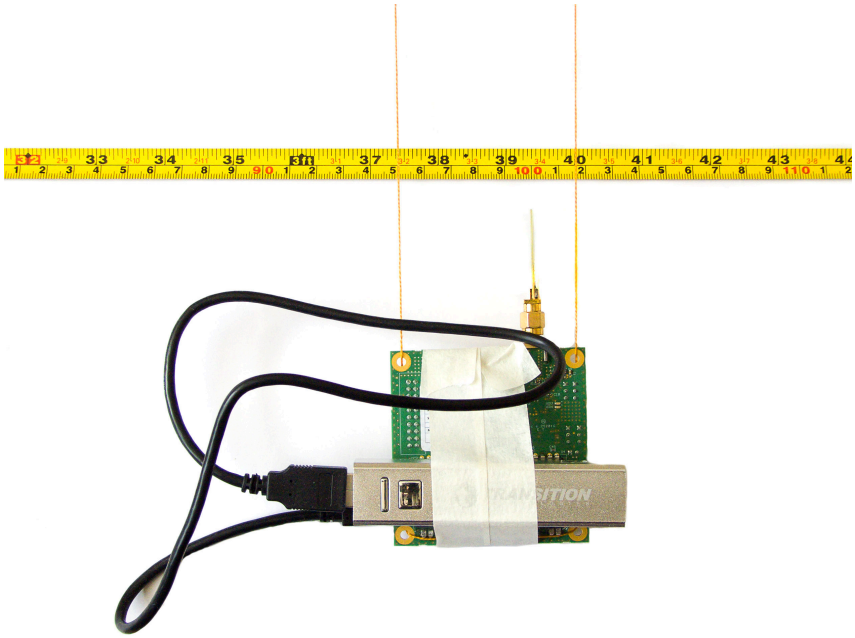


Fig. 4.10: The pendulum at rest, with the tape measure visible

$$\begin{cases} w(t) = A\mathrm{e}^{-\alpha t}\cos[2\pi f(t+t_0)] \\ x(t) = Q_x \\ y(t) = Q_y + w(t) \\ z(t) = Q_z + \sqrt{l^2 - w(t)^2} \end{cases} \tag{4.2}$$

8 measurements were made: 5 with the antenna perpendicular ($\perp$) to the direc-

tion of movement and 3 with it parallel (||). In the first stage, the damping coefficient α was determined from the maximum displacements in successive oscillations. The positions were read from the camera recording of the oscillations. The parameters of the exponential function were fitted to the collected points with methods from the NumPy library, using the default Levenberg–Marquardt algorithm. The fit of the damping in one of the measurements is shown in Figure 4.11.

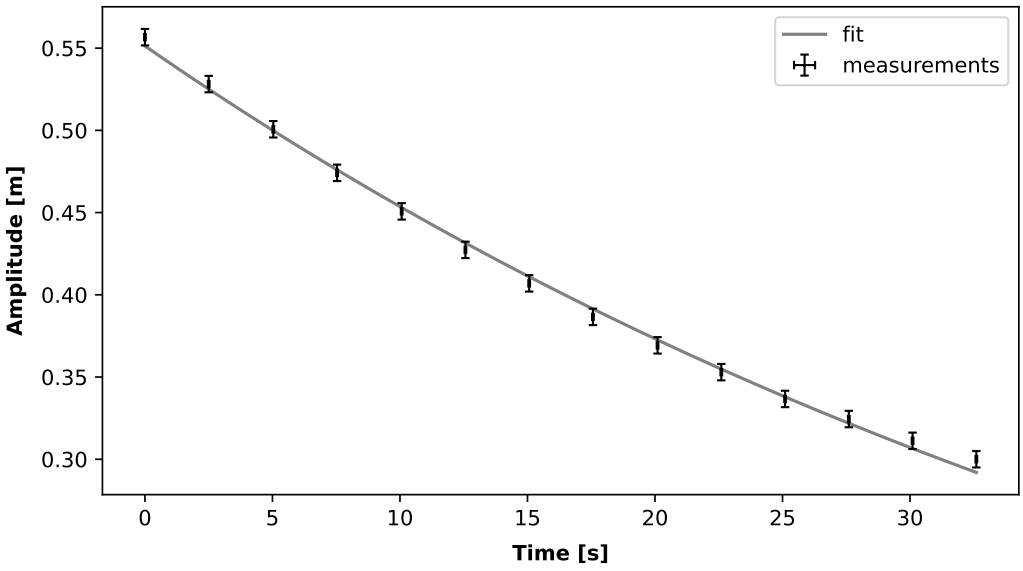


Fig. 4.11: Fit of the exponential damping in one of the measurements

Then, with a similar optimization using the oscillation function from (4.2), the delay t_0 of the system's measurements relative to the expected movement was determined and the points on the chart were fitted. Figure 4.12 shows three charts of the distances to each anchor measured during one of the tests, with the curve of the expected values. Table 4.4 gives the estimated measurement delay $\boldsymbol{t_0}$ and statistics such as the standard deviation $\boldsymbol{\sigma}$ from the expected curve and the number of samples in successive tests **T**, with the antenna orientation **O** relative to the direction of movement. The uncertainty of the delay $\boldsymbol{t_0}$ was taken as the duration of a single camera frame, i.e. 0.033 s (at 30 frames per second), as this limited how precisely the moments of the pendulum's extreme positions could be determined.

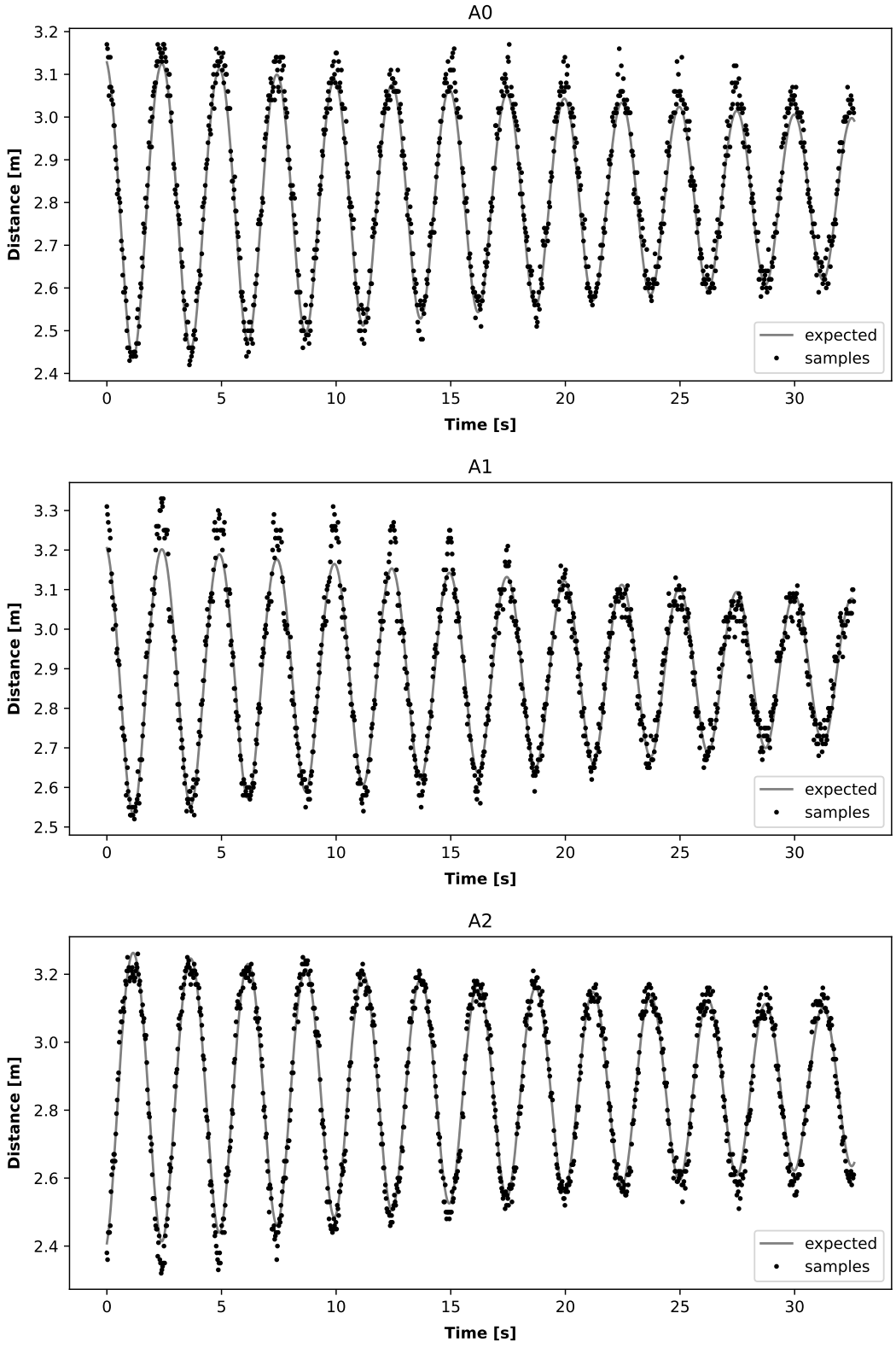


Fig. 4.12: Distances obtained in one of the pendulum tests

T	O	t_0 [s]	Samples	σ [m]		
				A_0	A_1	A_2
1	∥	(0.140 ± 0.033)	1072	0.032	0.042	0.041
2	∥	(0.090 ± 0.033)	902	0.036	0.041	0.034
3	∥	(0.106 ± 0.033)	1073	0.037	0.040	0.039
4	∥	(0.114 ± 0.033)	500	0.035	0.044	0.037
5	∥	(0.096 ± 0.033)	1069	0.043	0.047	0.040
6	⊥	(0.106 ± 0.033)	1469	0.034	0.050	0.068
7	⊥	(0.122 ± 0.033)	744	0.036	0.050	0.066
8	⊥	(0.179 ± 0.033)	1476	0.037	0.051	0.072

Tab. 4.4: Results of the dynamic measurements

4.4. Subjective tests

In the final stage, the whole system was tested in a real environment. A TP-Link WR-340G router was used for communication, as the access point of a Wi-Fi network to which only the devices needed for the tests were connected. The local network had no Internet access. This eliminated delays and errors caused by other devices communicating and by excessive numbers of packets on the network. The server application (described in Section 3.2) ran on a Lenovo ThinkPad X201 laptop with Linux Mint 17.1 Rebecca. The mobile device placed in the virtual reality headset was a Sony Xperia M4 with Android 5.0. The test was recorded on camera, together with recordings of the screens of both devices. The simulation ran at 50 frames per second on average, but enabling screen capture lowered this to about 30 FPS. An edited video showing part of the test is available at `https://youtu.be/IRgKCKVc36g` (accessed 05.09.2018). A single frame is shown in Figure 4.13.

This test also checked the alpha-beta filter described in Section 3.3. When the filter is turned off, the measurement uncertainty, in the form of position noise, is visible to the user as strong shaking. This causes so much discomfort that the system becomes unusable. The standard deviation of these errors can exceed 10 cm in some parts of the room. For this reason, without the support of an additional positioning system (for example an inertial one), filtering is essential for comfortable use of the system. The test showed that the filter introduces delays of several hundred mil-



Fig. 4.13: Frame from the video of the subjective test

liseconds and limits the maximum speed of movement to walking pace. Adjusting the filter coefficients can reduce these drawbacks, at the cost of some of the shaking coming back.

4.5. Power consumption test

During the tests, an attempt was made to estimate the energy consumption of the system's devices. One of the anchors was powered by four Ni-MH cells. Before the test, they were fully charged with an everActive NC-3000 charger[(2)] at 600 mA each, and their capacity C_{full} was then measured with a discharge current of 300 mA. After they had been fully charged again, a 5-hour session of the tests described above was carried out. The remaining charge C_{left} was then measured by discharging the cells in the charger to 0.9 V at 300 mA, and the charge used, C_{used}, was calculated. The approximate internal resistance R_{int} was also measured. The results are listed in Table 4.5.

[(2)] *everActive NC-3000 charger*: `http://everactive.pl/ladowarka-everactive-nc-3000` (accessed 05.09.2018)

ID	C_{full} [mAh]	C_{left} [mAh]	C_{used} [mAh]	R_{int} [mΩ]
0	2107	501	1606	225
1	2088	466	1622	240
2	2093	391	1702	246
3	2120	460	1660	248

Tab. 4.5: Results of measuring the energy consumption of a system device

The charger's manufacturer gives no detailed specification of how accurately it measures the charge or the internal resistance of the cells. Moreover, the exact dependence of the battery voltage on time was not examined. The result is therefore only approximate: the measurement uncertainty was taken as ±10 %, and the average voltage of the four cells as 5 V. The final result was about (330 ± 33) mAh, which corresponds to a power demand of the order of (1.6 ± 0.2) W.

Chapter 5

Conclusions

The system developed in this master's thesis is an example of a mixed reality (MR) application made up of several layers that run on different devices and work together. It made it possible to test to what extent a real-time locating system (RTLS) based on ultra-wideband (UWB) radio waves can translate the user's position into a virtual world generated by a mobile device. While the solution was being developed, numerous tests were carried out to check objectively whether the system is suitable for mixed reality applications. The first step was to examine whether the anchors can communicate with each other and measure the distances between them. This makes it possible to determine their coordinates in the room automatically. The results show that the system can determine the relative positions of the devices with sufficient accuracy (to within a few cm), which removes the need to set their positions in the application by hand. The user only has to specify the orientation of the anchors relative to the cardinal directions, so that the system works correctly with the compass, and the height at which they are mounted. In a final product, entering the rotation into the application by hand could be replaced by a visual calibration with the phone's camera. The user, standing inside the triangle formed by the anchors, would point the center of their field of view at the main anchor. Knowing the user's current orientation and the compass reading of the mobile device in the VR headset, the system could then calculate the required rotation angle.

The static tests (described in Section 4.2) made it possible to determine how the system's readings deviate depending on the user's position in the room. The most

accurate results were obtained in the central part of the triangle formed by the anchors (Figure 4.6). Closer to the walls or to the devices themselves, the uncertainty of the coordinates grows. For this reason, it is worth mounting the anchors so that they cover as much of the room as possible and the user is never closer than 1–2 m to them, and in particular never directly beneath them. However, the manufacturer's recommendation to place the anchors at least 15 cm from the nearest wall or metal object should be kept in mind. The tests showed that the orientation of the antenna affects the measured distances (Figure 4.8). The differences usually did not exceed a few cm, which means that the error is practically unnoticeable in the application. An antenna with a more omnidirectional radiation pattern might eliminate this problem. An alternative would be to correct the location based on the user's orientation obtained from the compass.

The tests also showed that the lack of direct line of sight between the transmitters (NLOS) disturbs the measurements. A person blocking the shortest line between the tag and an anchor changes (usually increases) the distance readings by a dozen or so centimeters (Figure 4.9). The devices must therefore be mounted so that there are no obstacles in the communication path. A proven solution is to attach the anchors to the ceiling or to the side walls, above the maximum height reached by the system's moving devices, while the tags should be above the users' heads. Moreover, this kind of interference suggests that radio localization is poorly suited to tracking the position of handheld controllers or of individual limbs. These will usually be below the head, frequently blocking the communication with the anchors and so causing incorrect position readings. A partial solution to this problem could be algorithms that correct the readings of devices that, according to geometric calculations, are probably blocked.

The dynamic pendulum tests (Section 4.3) were meant to simulate fast movement of the user and determine how accurately the position is determined under such conditions. The delay t_0, measured from the moment the tag is at a known position until its coordinates are obtained (given for each test in Table 4.4), averaged (0.109 ± 0.033) s. This is somewhat more than the expected 0.030 s that results from the way the DecaWave system processes the measurements and communicates

between its devices[1]. The difference may come from the processing of the measurements by the system's devices, in particular by the anchor that is connected to the computer and collects the data from the other modules. Further delays may be introduced by the UART buffers on both the DecaWave device and the computer side, and by the Qt library that handles the serial port in the server application. The accuracy of $\boldsymbol{t_0}$ was also affected by the method used, which determined the times of the pendulum's maximum displacements from the camera recording. The distance charts in Figure 4.12 show clear disturbances at the moments of maximum displacement. They may come from additional vibrations of the pendulum when it changed direction, including a partial rotation about its own axis, which changes the measured distance, as the earlier tests showed. Apart from these disturbances, the position accuracy during movement does not differ noticeably from that at the static points.

The subjective tests (Section 4.4) made it possible to test the whole system in a real room, and so to determine whether the solution is suitable for use in a final product and whether it is easy and intuitive enough for the end user. To make the system more responsive, an additional localization mechanism could be used, for example an inertial one (IMU). This would improve the response to small movements, while the gyroscope drift and the accelerometer noise would be corrected by the DecaWave system. Similar solutions are used in the mixed reality localization systems on the market, which combine imprecise large-scale mechanisms with more accurate inertial ones that are prone to drift. With a mobile device such as a smartphone, its own sensors could be read and used to improve the tracking of the user's position. Of course, dedicated IMU modules, such as the one in the Samsung Gear VR headset, would offer much greater positioning accuracy and smoothness.

It should be noted that the developed system is a prototype, and some of its elements would be changed before it became a product ready for sale. One of them is the design of the VR headset worn by the user. A good solution would be to integrate the tag and its power supply into the headset, and in particular to miniaturize the

[1] The delay arises because the measured distance is only received in the next superframe. The details are described in Section 3.1.

integrated circuits and the antenna. DecaWave recommends small surface-mount UWB antennas that it has tested [20]: Taiyo Yuden AH 086M555003 or Partron ACS5200HFAUWB. Moreover, since the anchors are usually above the tags, the radiation patterns of the tags' antennas could be adjusted so that the strongest signal is transmitted upwards from the user's head. This would both reduce the tags' energy consumption and reduce the radiation towards the person using the system. Another element that has to be worked out before the system becomes a finished product is how the anchors are mounted and powered. With suspended ceilings, the devices could be attached to the ceiling grid. Another option would be to hang them on brackets fixed to the walls. A more portable solution, where neither is possible, would be to place the anchors on high shelves or on tripods. The devices could be powered by cables running to a power supply, or by a built-in rechargeable battery that would need charging every few to a dozen or so hours of operation, depending on its capacity.

The system could also be scaled up to a larger area by adding more anchors in the room. This would require redesigning how the devices communicate with each other and connecting them to a central managing computer [8]. That computer would determine which anchors are closest to a given user and can determine their position in space most accurately. More stationary devices would also help, to some extent, with the communication path to the tags being blocked. Moreover, more users could then use the system at the same time.

Bibliography

[1] P. Milgram, F. Kishino: *A Taxonomy of Mixed Reality Visual Displays.* In: *IEICE Trans. Information Systems*, vol. E77-D, no. 12, 12.1994, https://www.researchgate.net/publication/231514051_A_Taxonomy_of_Mixed_Reality_Visual_Displays (accessed 05.09.2018)

[2] J. Jerald: *The VR Book: Human-Centered Design for Virtual Reality*, ACM and Morgan & Claypool, New York, USA, 16.10.2015.

[3] P. Lawitzki: *Application of Dynamic Binaural Signals in Acoustic Games*, Stuttgart Media University, 02.03.2012, http://plaw.info/wp-content/uploads/MasterThesis_Lawitzki_2012.pdf (accessed 05.09.2018)

[4] P. Lawitzki: *Android Sensor Fusion Tutorial*, 2014, http://plaw.info/articles/sensorfusion/ (accessed 05.09.2018)

[5] Microsoft HoloLens: *The World's First Holographic Head-Mounted Display*, https://www.microsoft.com/en-us/hololens/ (accessed 05.09.2018)

[6] IEEE: *Std 802.15.4-2011: Standard for Local and metropolitan area networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 05.09.2011, https://standards.ieee.org/getieee802/download/802.15.4-2011.pdf (accessed 05.09.2018)

[7] DecaWave: *TREK1000 User Manual*, Version 1.06, 2016, https://www.decawave.com/system/files/resources/trek1000_user_manual_1.06.pdf (accessed 05.09.2018)

[8] DecaWave: *APS016 Application Note: Moving from TREK1000 to a product*, Version 2.01, 2015, https://www.decawave.com/sites/default/files/aps016_moving_from_trek1000_to_a_product.pdf (accessed 05.09.2018)

[9] DecaWave: *DecaRangeRTLS ARM Source Code Guide*, Version 2.1, 30.10.2015, (not available online)

[10] DecaWave: *DecaRangeRTLS PC Source Code Guide*, Version 2.1, 30.10.2015, (not available online)

[11] DecaWave: *DW1000 EVB*, Revision 0-03, 19.07.2013, (not available online)

[12] STMicroelectronics: *STM32F105xx STM32F107xx Datasheet*, (DocID15274 Rev 10), 03.2017, http://www.st.com/resource/en/datasheet/stm32f105rc.pdf (accessed 05.09.2018)

[13] STMicroelectronics: *STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx Reference Manual*, RM0008, (DocID13902 Rev 17), 08.2017, http://www.st.com/resource/en/reference_manual/cd00171190.pdf (accessed 05.09.2018)

[14] Android Developers: *Motion sensors*, 24.04.2018, https://developer.android.com/guide/topics/sensors/sensors_motion (accessed 05.09.2018)

[15] Android Developers: *Sensors Overview*, 24.04.2018, https://developer.android.com/guide/topics/sensors/sensors_overview (accessed 05.09.2018)

[16] Android Developers: *Create an Android Library*, 08.05.2018, https://developer.android.com/studio/projects/android-library (accessed 05.09.2018)

[17] Unity - Manual: *Building and using plug-ins for Android*, 18.05.2017, https://docs.unity3d.com/Manual/PluginsForAndroid.html (accessed 05.09.2018)

[18] R. Penoyer: *The Alpha-beta Filter*, 07.1993, http://collaboration.cmc.ec.gc.ca/science/rpn/biblio/ddj/Website/articles/CUJ/1993/9307/penoyer/penoyer.htm (accessed 05.09.2018)

[19] J. H. Painter, D. Kerstetter, S. Jowers: *Reconciling steady-state Kalman and alpha-beta filter design*, IEEE Transactions on Aerospace and Electronic Systems, 11.1990, https://ieeexplore.ieee.org/document/62250/ (accessed 05.09.2018)

[20] DecaWave: *APH0007 Application Note: Antenna Selection / Design Guide for DW1000*, Version 1.0, 2014, https://www.decawave.com/content/antenna-selectiondesign-guide-dw1000 (accessed 05.09.2018)

List of Figures

2.1. Classification of augmented and virtual realities 9
2.2. Diagram of IMU sensor data fusion 10
2.3. Google Cardboard VR viewer 11
2.4. Samsung Gear VR headset with its controller 13
2.5. Photo taken with the Sony AR effect app 14
2.6. The Microsoft HoloLens device 15
2.7. HTC Vive Pro with controllers and Lighthouse base stations 16
2.8. Oculus Rift with Touch controllers and Constellation cameras 18
2.9. InterSense IS-900 device with CrystalEyes glasses 20

3.1. DecaWave EVB1000 evaluation board 23
3.2. Data exchange in a single slot 24
3.3. ST-Link/V2 programmer used to program the EVB1000 boards 26
3.4. The server application 28
3.5. Geometry of anchor autopositioning 29
3.6. Graphical representation of tag trilateration 31
3.7. The VR BOX 2.0 headset used in this thesis 35
3.8. Bluetooth remote from the VR BOX 2.0 set 36
3.9. The scene in the Unity editor 37
3.10. Smartphone sensor coordinate system in Android 38
3.11. QR code with the configuration of the VR BOX 2.0 headset 38
3.12. Screenshot of the mobile application on the phone 39
3.13. User wearing the headset with the EVB1000 tag 40

4.1. Anchors mounted for the tests 42
4.2. Layout of the anchors and the static measurement points in the room 43
4.3. Histograms of the autopositioning distances 44
4.4. Tripod with the tag during the static measurements 46
4.5. Weight indicating the exact position of the tripod 47
4.6. Map of the reading accuracy at the static points 48
4.7. Position accuracy as a function of the distance from the anchor 49
4.8. Distance readings at point E depending on the antenna orientation .. 49
4.9. Effect of blocking the transmitters on the distance readings 50
4.10. The pendulum at rest, with the tape measure visible 51
4.11. Fit of the exponential damping in one of the measurements 52
4.12. Distances obtained in one of the pendulum tests 53
4.13. Frame from the video of the subjective test 55