FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Oskar Razyapov

Smart extensions to regular cameras in the industrial environment

Department of Software Engineering

Supervisor of the master thesis:	Mgr. Ladislav Peška, Ph.D.
Study programme:	Computer Science
Study branch:	Software and Data Engineering

Prague 2024

I would like to express my heartfelt gratitude to my supervisor, Prof. RNDr. Peter Vojtáš, DrSc., for his endless support during my journey. Additionally, I thank to my friend, RNDr. Štěpán Balcar. They provided me with advice during the preparation of this thesis, and their patience and help during the experiments were invaluable. I also thank them for providing me with additional equipment and for the friendly lunches that kept me motivated.

I extend my sincere thanks to Mgr. Ladislav Peška, Ph.D., for his careful review of the thesis and support before submission.

Title: Smart extensions to regular cameras in the industrial environment

Author: Oskar Razyapov

Department: Department of Software Engineering

Supervisor: Mgr. Ladislav Peška, Ph.D., Department of Software Engineering

Abstract: The purpose of this work is to obtain an extensive set of videos with automatically annotated locations of people. Using the collected data, we have developed a Pixel-to-Real machine learning model, which aims to extend static Closed-circuit television (CCTV) cameras to allow for localization of people and measuring distance between the them, solely based on the input video data. To gather the datasets, we have developed an Ultra-wideband (UWB) system, which allows for precise people localization. By synchronizing UWB and video data, we generated automatically annotated datasets, which were used for training the Pixel-to-Real model. To facilitate the data acquisition, we have developed a GUI, which allows to synchronize, visualize, and analyze video and UWB data for fast and easy system calibration. Additionally, it facilitates an export of the data for the model training. To demonstrate the accuracy of the developed model, we have developed the Optical method for people localization based on the height of people, and compared it with Pixel-to-Real model. We also performed a comprehensive evaluation of UWB, Pixel-to-Real, and Optical methods against Ground Truth people positions. Our methodology ensures the repeatability of the experiments, which allows us to support the future research and development in people localization.

Keywords: repeatable experiments, cctv cameras, localization, ultra-wide band, machine learning

Název práce: Chytré extenze jednoduchých kamer v průmyslovém prostředí

Autor: Oskar Razyapov

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Ladislav Peška, Ph.D., Katedra softwarového inženýrství

Abstrakt: Účelem této práce je získat rozsáhlou sadu videí s automaticky anotovanými pozicemi lidí. Na základě shromážděných dat jsme vyvinuli model strojového učení Pixel-to-Real, jehož cílem je vylepšit statické kamery uzavřeného televizního okruhu (CCTV) tak, aby umožňovaly lokalizaci lidí a měření vzdáleností mezi nimi pouze na základě vstupních video dat. K získání datasetů jsme vyvinuli ultraširokopásmový (UWB) systém, který umožňuje přesnou lokalizaci lidí. Synchronizací UWB a video dat jsme vygenerovali automaticky anotované datasety, které byly použity pro natrénování modelu Pixel-to-Real. Pro usnadnění sběru dat jsme vyvinuli GUI, které umožňuje synchronizaci, vizualizaci a analýzu video a UWB dat pro rychlou a snadnou kalibraci systému. GUI také usnadňuje export dat připravených pro natrénování modelu. Pro demonstrace přesnosti vyvinutého modelu jsme vyvinuli metodu Optical, která je založena na výšce osob, a porovnali ji s modelem Pixel-to-Real. Provedli jsme také komplexní hodnocení metod UWB, Pixel-to-Real a Optical oproti reálným (Ground Truth) pozicím lidí. Naše metodologie zajišťuje opakovatelnost experimentů, což podporuje budoucí výzkum a vývoj v oblasti lokalizace lidí.

Klíčová slova: opakovatelné experimenty, cctv kamery, lokalizace, ultra širokopásmová technologie, strojové učení

Contents

1 Introduction

Data annotation is a critical part of the training a machine learning model. Unfortunately, it still has to be done manually, which is both time-consuming and prone to human errors. The automation of this process can significantly reduce annotation time, minimize costs and increase data quality.

1.1 Motivation

The primary objective of this work is to proof the concept of automatic data annotation and acquire an extensive set of videos with automatically annotated locations of people.

To demonstrate the usability of the collected data, we have developed a machine learning model, which allows to predict people's locations and measure distance between them, solely relying on the input video streams. This model is aimed to extend the capabilities of traditional static and indoor Closed-circuit television (CCTV) cameras.

The main use cases of our system can be:

- **Covid-19 social distancing**: to ensure safe distances are maintained in public spaces, e.g. schools, offices, and shopping centers.
- **Hazardous work environments**: to track the workers' locations in industrial environments, where there is a risk of contact with hazardous materials or dangerous equipment.
- **Indoor navigation and safety**: to assist in navigation in large indoor spaces such as hospitals, airports, and shopping malls. Localization data can improve emergency response times, and help with optimization of environmental conditions.
- **Surveillance and security**: to identify unauthorized access or suspicious activities.
- **Climate Control Systems**: to adjust heating, air-conditioning, and lighting based on real-time occupancy data. This allows to improve energy efficiency and save money.
- **Asset tracking**: to track movement of an equipment and its localization in industrial environments and warehouses.
- **Assistance to people with disabilities**: to assist people with vision impairments. Our system can enhance safety by identifying objects in the immediate vicinity of a person. This can be achieved through the integration of an additional sensor that informs people about obstacles, for example, in front of them.

We propose a cost-effective solution that solely relies on the existing infrastructure consisting of CCTV cameras and a PC acting as a server. This system makes advanced positioning capabilities accessible to a broader range of users.

1.1.1 IoT and Industry 4.0

Our system uses Internet of Things (IoT) technologies. This allows us to achieve more precise positioning in larger areas of indoor environments. IoT technologies enable devices to communicate over the internet, allowing real-time data collection, which enhances the capabilities of existing surveillance systems. The IoT sensors provide additional contextual data, which improves the accuracy and reliability of the surveillance. This integration supports the principles of Industry 4.0, promoting automation and inter-connectivity in surveillance systems, allowing for more efficient management of indoor environments.

1.2 Our work

The development of a machine learning model requires extensive datasets with accurate annotations. However, publicly available data often have low quality [1]. To address this, we created our own annotated dataset. Additionally, we used localization sensors to improve the accuracy in location estimates.

To gather accurate people localization data for training the model, we have developed own Ultra-wideband (UWB) network. The reason for choosing UWB is described in Chapter 2. The UWB technology allows to accurately determine the location by calculating time it takes to signals to travel between transmitting and receiving devices. A detailed description of UWB technology is provided in Section 2.1.2.

To ensure the precision of the collected data a considerable time was spent collecting and analyzing the data. Therefore, to facilitate the data acquisition, we have developed a Graphical User Interface (GUI), which is called the *Indoor Positioning System*. This application enables us to synchronize, visualize, and analyze video and UWB data, allowing for fast and easy system calibration. This synchronization generates accurate, automatically annotated datasets, essential for training the machine learning model.

Our methodology ensures the repeatability of the experiments, which allows to support the future research and development in people localization. The *Indoor Positioning System* (GUI) helps developers to check the precision of their localization methods or develop new ones based on our dataset.

To demonstrate the practical application of our collected datasets, we have developed two distinct methods for people localization in indoor environments:

- **Pixel-to-Real method**: a predictive model trained on a multi-tag UWB system synchronized with video data. This is the model we aim to train.
- **Optical method**: a geometric people localization approach, which utilizes camera's intrinsic parameters and height of observed people. This approach does not require the collected UWB annotations, but rather relies only on the video input. This method is developed solely to compare it with Pixel-to-Real model and show that our *Indoor Positioning System* application can help evaluate any people localization method.

In general, we have compared and evaluated the Pixel-to-Real model against both Optical and UWB localization methods. For all methods, we used our *Indoor*

Positioning System application to extract the estimated people locations. The comparison is detailed in Chapter 7.

An example of our application is shown in Figure 1.1.

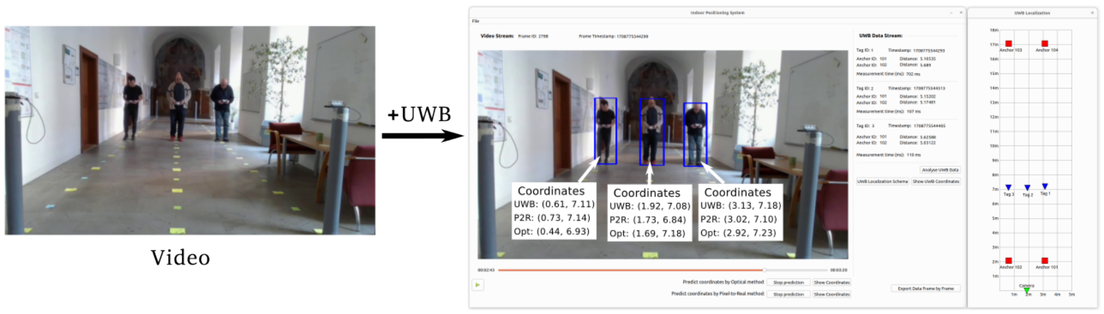


Figure 1.1 An example of how video and UWB data can be synchronized and visualized in *Indoor Positioning System*.

1.2.1 System installation process

With the *Indoor Positioning System* application our localization system can be quickly installed in the customer's premises.

The installation process of the localization system is comprised of the following steps:

- **Input data stream**: In the customer's premises, we acquire a continuous stream of input data from their CCTV cameras.
- **Data acquisition for system calibration**: We then collect the UWB measurements on the server for further analysis and calibration of our system.
- **Calibration**: Once the data is collected, the system undergoes a detailed calibration process. This post-processing step involves aligning the UWB data with the video data to ensure the synchronization of both data streams.
- **Model training**: The calibration process produces annotated data that combines video frames with precise location information. This annotated dataset is then used to train the Pixel-to-Real model, which learns to predict real-world coordinates from pixel coordinates, enabling the system to perform accurate localization based on visual input alone.
- **Deployment of the model**: Once trained, the Pixel-to-Real model is deployed within the customer's site. The customer obtains a fully automated localization system that operates based on the pre-trained model, requiring no further intervention. This allows to use advanced positioning capabilities without needing needing a technical knowledge, expertise, or expensive additional hardware.

During the deployment, the system calibration is performed in a controlled manner, using our own wireless network for the communication between the UWB devices and server. This approach avoids potential privacy restrictions and ensures that our system does not affect the existing customer's infrastructure. After deployment, the customer will be able to use the *Indoor Positioning System* application. However, critical functions will be restricted to qualified personnel and developers.

1.2.2 Experiments and challenges

Throughout the work, a variety of experiments have been performed in different environments to simulate real-world scenarios and test the robustness of the developed methods. These environments include a light manufacturing setup with obstacles like cabinets and electrical enclosures, a long obstacle-free hallway, and a laboratory equipped with computers.

During the experiments we encountered several challenges, including signal interference, data shifts, and synchronization issues. By implementing techniques such as rolling standard deviation and polynomial regression, we improved the accuracy of UWB distance measurements. Additionally, using timestamps of recorded data, we ensured reliable synchronization between UWB and video data.

1.3 Thesis Structure

The thesis is structured as follows. Chapter 2 discusses existing methods for people localization, highlighting their weaknesses that our work is aimed to resolve. Chapter 3 describes our journey in developing our own UWB network, including the issues that we have encountered and resolved. Chapter 4 demonstrates how the synchronization between video and UWB data is performed to create automatically annotated datasets. Chapter 5 details the training process of the Pixel-to-Real model.Chapter 6 explains the Optical method, including the camera calibration process. Chapter 7 provides a comprehensive evaluation and comparison of UWB, Pixel-to-Real and Optical localization methods, highlighting their strengths and weakness. Finally, Chapter 8 concludes the work.

Two appendices are included: Appendix A describes notations used throughout the work, which facilitates the understanding of the conducted experiments and the collected data. Appendix A contains a list of figures.

2 Related Work

During the research, we explored different methodologies for indoor people localization. For each method, we have highlighted their disadvantages and the gaps that our work aims to address.

Several techniques may be used for indoor people localization, including Wi-Fi [2], Bluetooth Low Energy (BLE) [3], Radio Frequency Identification Device (RFID) [4], Ultra-wideband (UWB) [5], Computer vision [6] and Neural networks [7]. All these methods aim at calculating the distances between static and dynamic objects. The coordinates of static objects are known beforehand, while the coordinates of dynamic objects are determined based on the distances provided by the above techniques.

2.1 Comparison of localization techniques

Research performed by Zafari, F. and Gkelias, A. and Leung, K. [8] discusses the capabilities and limitations of different localization techniques, emphasizing their accuracy in short-range environments. This work states that **precise localization systems often require extra hardware**, which might be expensive, especially for small organizations. Additionally, it mentions the **lack of standardization and benchmarking** for existing people localization techniques.

2.1.1 Received Signal Strength Indicator (RSSI)

RSSI is a measurement of the power present in a received radio signal. It is used to estimate the signal strength and then determine the distance between devices [9].

Several studies explore the use of radio frequency localization techniques, which utilize the **RSSI**. These techniques include the use of BLE and Wi-Fi technologies.

The BLE is a wireless communication technology designed for low power consumption and short bursts of data transfer, compared to classic Bluetooth. The BLE is used for people localization by measuring RSSI between devices.

While the BLE and Wi-Fi approaches are simple and cost-effective, they have low localization accuracy, especially in Non-line of Sight (NLoS) conditions. This limitation appears primarily due to RSSI fluctuations caused by multi-path fading, when signals and their reflections arrive at the receiver in different paths, resulting in interference that impacts the received signal strength [10].

Moreover, the accuracy is further compromised by signal attenuation caused by its passage through walls and other substantial obstacles. This problem is explored in the work of Zafari, F. et al. [8]. It is worth mentioning that industrial premises are often polluted by signals from a large number of other hardware devices, which causes all these limitations.

Additionally, BLE devices operate on the overcrowded 2.4 GHz ISM band [11], which is also used by classic Bluetooth [12] and Wi-Fi technology. This overlap causes signal interference and multi-path fading, affecting the work of existing infrastructure.

2.1.2 Ultra-Wideband

Ultra-wideband (UWB) is a type of wireless communication technology that uses a wide frequency spectrum (typically from 3.1 to 10.6 GHz) to transmit signals (data) between UWB devices. The distance between devices is calculated based on the time (Time of Flight (ToF)) it takes for the signal to travel from the transmitting device to the receiving device. The communication protocol used in this technology is described in detail in Section 3.3.1.

Compared to the BLE and Wi-Fi, UWB operates on a wider range of frequencies, utilizes short-duration pulses for communication, and has low power spectral density, allowing for more efficient elimination of signal interference [13].

Furthermore, recent research shows that, in general, UWB technology outperforms BLE v5.1 in complex environments with obstructed Line of Sight (LoS) and significant multi-path effects [14].

2.1.3 Computer vision

The computer vision techniques do not require the use of any additional hardware. However, they often have lower localization accuracy compared to other hardware. The most common computer vision techniques include bird's-eye view [6], the use of people heights and stereo vision [15].

Stereo vision

The stereo vision is one of the most accurate computer vision method for people localization [15]. It involves the use of two or more identical cameras placed at different angles to capture images of the same scene from different perspectives. By comparing the differences between the captured images, it is possible to calculate the depth information and determine the precise location of people in an environment [16].

Unfortunately, this method has a significant downside — it requires all cameras to be perfectly aligned in order to precisely calculate the distance to the observed object. This alignment involves ensuring that the cameras are correctly positioned and that their fields of view overlap accurately, which allows for precise triangulation. This calibration process is very time-consuming. Furthermore, this requires a customer to buy additional hardware, and potentially change the existing CCTV camera infrastructure, which is very expensive.

Bird's-eye view

The bird's-eye view refers to a perspective that looks down on an area from above. The goal is to make the scene flat, so that all objects (in the image) appear at the same depth. In this representation, it is possible to calculate the real-world coordinates of people using their x and y pixel coordinates from the image. The downside of this approach is that it significantly relies on the angle of view of the camera and requires the transformation process of the normal camera view to a bird's-eye view. This process itself involves significant errors in distance estimation.

Geometric approach

Given these considerations, we have implemented a geometric-based people localization, which utilizes the height of each person in the image to estimate their locations. Later in our work, this method is referred to as the Optical method. In Section 7.4, we will reveal that the Optical method has lower accuracy compared to the UWB and developed Pixel-to-Real method.

2.2 Addressing the above mentioned problems

These existing methodologies often lack the capability for automatic annotation, which is crucial for training and validating machine learning models. Our solution is aimed to addresses this gap by providing automatically annotated video data.

Moreover, most people localization techniques require additional hardware. To address this, we propose a cost-effective solution (the Pixel-to-Real model) that solely relies on the existing infrastructure.

Based on all the advantages of UWB technology over other IoT technologies, we decided to implement this technology in our work.

3 Study of UWB technology: calibration and experiments

This section is intended solely to provide a detailed description of communication between UWB devices and the challenges encountered while attempting to implement an UWB localization.

Initially, we planned to use an existing implementation of the UWB localization. Based on the collected UWB data, we wanted to synchronize it with video recordings for training the Pixel-to-Real model.

However, we have not found any existing implementation of UWB localization that would suit our goals. Therefore, we decided to implement it from scratch.

3.1 ESP32 UWB board

We chose the ESP32 board from Makerfabs[1], equipped with a BU01 UWB module[2], which is based on the Decawave's UWB DW1000[3] module. An example of the ESP32 board integrated with an UWB module is depicted in Figure 3.1.

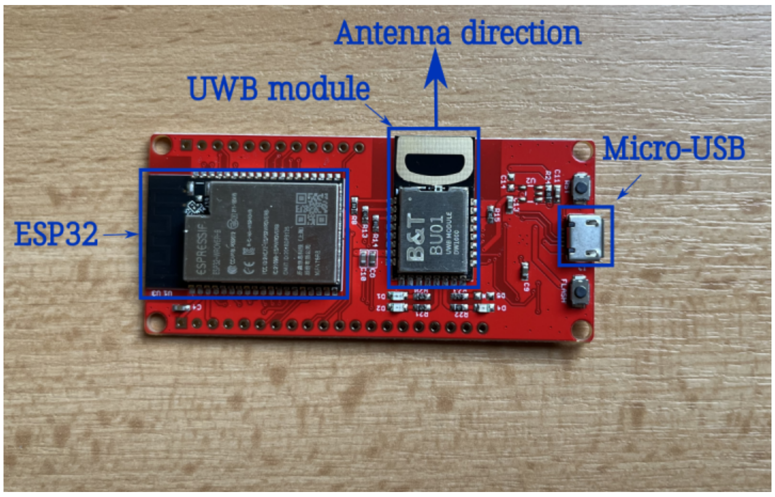


Figure 3.1 ESP32 UWB board from Makerfabs. Please pay attention to the antenna direction, which is crucial in further experiments. The board has a *microUSB* port, which is used to power the board, as well as to connect it to a laptop.

The ESP32 board is a low-cost and low-power System-on-a-Chip (SoS) that includes a microprocessor, Wi-Fi, and dual-mode Bluetooth, which enable a reliable communication over the wireless network. The BU01 UWB module further enhances this capability by enabling a communication between UWB chips to calculate precise locations.

The ESP32 board integrated with UWB module can be of two types: an anchor and a tag. The specific role is determined by different firmware versions uploaded onto the device, enabling the same hardware to perform various tasks. An *anchor*

[1]https://www.makerfabs.com/esp32-uwb-ultra-wideband.html

[2]https://docs.ai-thinker.com/_media/uwb/docs/bu01_product_specification_en_v1.0.pdf

[3]https://www.qorvo.com/products/p/DW1000

acts as a stationary reference point, essential for precise position determination. On the other hand, a *tag* acts as a mobile device attached to an object whose position need to be tracked.

The UWB module performs continuous scanning to lock onto another UWB module. Upon establishing a connection, they initiate a ranging process, which measures Time of Flight (ToF) for data packets to travel back and forth between the boards. This time is then multiplied by the speed of light to provide the actual distance between boards [17] (the communication protocol is described in Section 3.3.1 in more detail). The calculated distance is then used to determine the location using triangulation[4]. This technique is discussed in detail in Section 3.4.5.

In addition, the BU01 UWB module is equipped with an antenna that is used to transmit the signal, as shown on Figure 3.1. Please note the importance of the antenna orientation, as it plays a crucial role in the performance and accuracy of our further experiments. Correct antenna alignment is important for correct signal transmission and reception.

The ESP32 board requires an external power supply to operate, because it does not have a built-in battery. On the other hand, it is equipped with a *microUSB* port, which can be used to connect power banks. The power banks allow for flexible use of the boards without the need for a stationary power supply.

Firmware setup

The ESP32 board is designed to be compatible with the Arduino IDE[5], which simplifies the process of writing and uploading the firmware.

Before uploading the firmware, it is necessary to set up the Arduino IDE with the essential ESP32 packages, which enable to work with ESP32 boards. Upon successful installation of the necessary libraries, the firmware for both the anchor and tag can be uploaded to the boards.

3.1.1 Antenna calibration

The data packets transmitted from one device to another contain the time of transmission. Once the counterpart receives the data packet, it records the time of reception and calculates the ToF based on these two timestamps.

However, as shown in Figure 3.1, the UWB module is equipped with an antenna. The delay caused by this antenna also contributes to the resulting ToF. This antenna delay is internal to the module, and therefore, should not be included in the resulting ToF, as illustrated in Figure 3.2.

Therefore, an **accurate antenna calibration** is essential. This calibration involves adjusting the ToF to exclude the antenna delay. This step is critical for achieving accurate distance measurements, as even minor discrepancies can lead to significant errors in position estimation. For instance, a **3 ns error** in ToF can result in an approximately **1-meter error** in distance measurement due to the high speed of wireless signal propagation.

The calibration process is conducted using a known distance between the tag and anchor, performing iterative adjustments to the delay parameter until the

[4]https://en.wikipedia.org/wiki/Triangulation
[5]https://www.arduino.cc/en/software

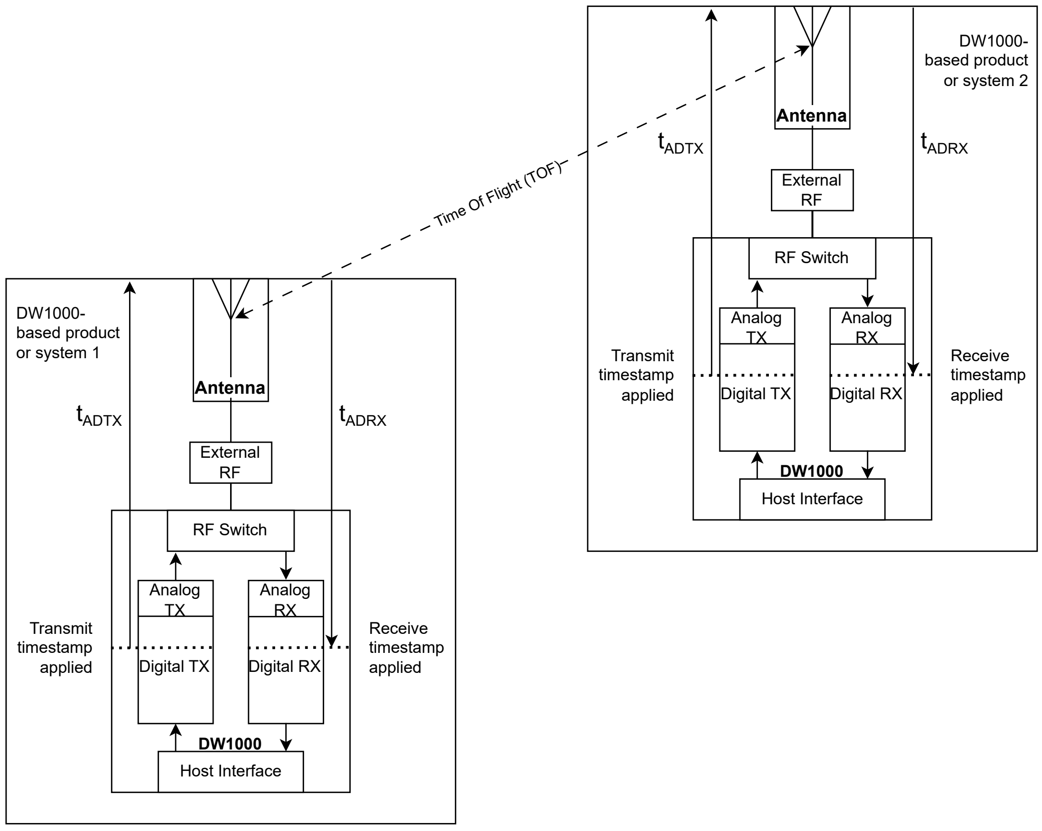


Figure 3.2 The antenna delay diagram [18].

measured distance closely matches the actual distance. This process is performed by a specific firmware designed for calibration purposes[6]. Due to variations in the antennas provided by the factory, each UWB device must be calibrated individually to ensure accuracy.

An example of the placement of the tag and the anchor during the calibration process is depicted in Figure 3.3, where the calibration distance is 1 meter. In our experiments, we established the actual reference distance using ground markers, and calibrated devices at a distance of 5 meters. The reason is discussed in Section 3.4.4. The official recommendations state that it is enough to set the antenna delay either on a tag or anchor. However, the results of our experiments indicate that in order to achieve accurate results, each tag and each anchor must be calibrated separately.

3.1.2 One Anchor – One Tag

After calibrating the antenna, we initiated a test run of the system with a single tag and a single anchor to evaluate the basic functionality and accuracy of the UWB ESP32 boards. For this, we used the existing implementation, provided by the official library[7].

We initiated a ranging process, where the anchor was continuously measuring the distance to the tag. We logged the results to the serial console in Arduino

[6]https://github.com/jremington/UWB-Indoor-Localization_Arduino
[7]https://github.com/thotro/arduino-dw1000/tree/master/examples

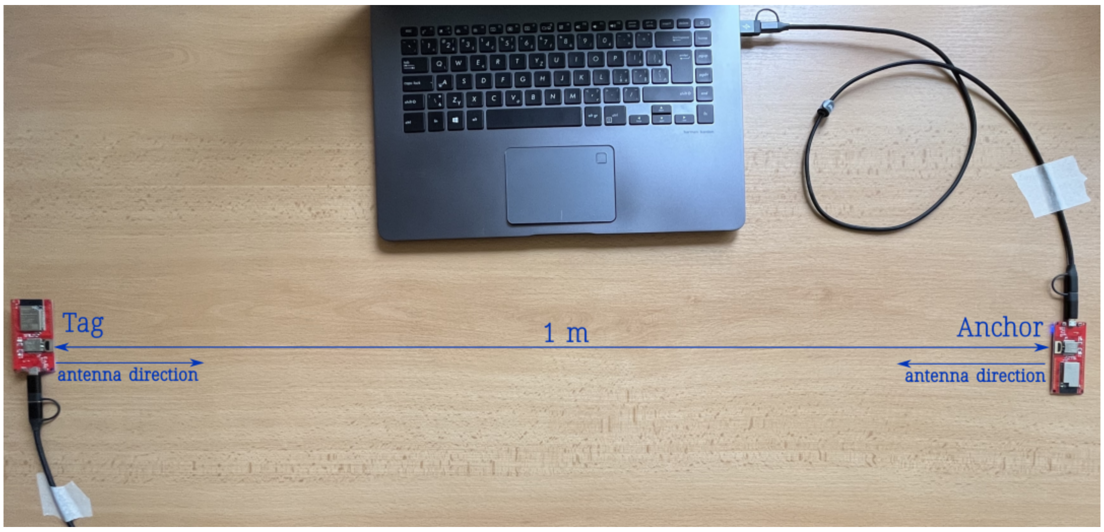


Figure 3.3 An example of the placement of the anchor and the tag during the antenna calibration process. Antennas of UWB modules are directed towards each other making a clear LoS. The antenna is calibrated at a distance of 1 meter. The anchor is connected to a laptop, enabling the observation of results logged to the serial console in the Arduino IDE. The tag is powered by a power bank.

IDE, providing real-time feedback on distance measurements and any potential anomalies observed during the experiment. This initial test run was essential to verify the system operational integrity before adding complexity and scaling up the number of tags and anchors.

During several test runs, we discovered that the boards communicate most effectively when UWB modules are directed towards each other, achieving a Line of Sight (LoS) communication. An example of the LoS communication is shown in Figure 3.3. Conversely, when UWB modules were directed away from each other, they performed a Non-line of Sight (NLoS) communication, as illustrated in Figure 3.4. This led either to signal loss or to an increase in estimated distances due to signal propagation effects, which were caused by reflection and diffraction of the signals from other objects. In the following text, these measurements are referred to as *"Towards Each Other"* and *"Away From Each Other"* measurements, respectively.

The difference in the *"Towards Each Other"* and *"Away From Each Other"* measurements is further illustrated in the box plot, depicted in Figure 3.5. The plot shows the spread and tendency of the both types of measurements at a distance of 1 meter. The *"Towards Each Other"* measurements are more tightly clustered around the mean, indicating less variability compared to the *"Away From Each Other"* measurements. Additionally, the histograms shown in Figure 3.6 reveal that the *"Away From Each Other"* measurements show a wider spread of values from the mean, and skewness towards higher values compared to the *"Towards Each Other"* measurements. This results in the higher standard deviation and variance. The statistics for these measurements are presented in Table 3.1.

An example of the collected data with antennas directed towards each other is shown in Table 3.2. This table shows the samples of measured distances in meters, which are calculated based on ToF of the signal.

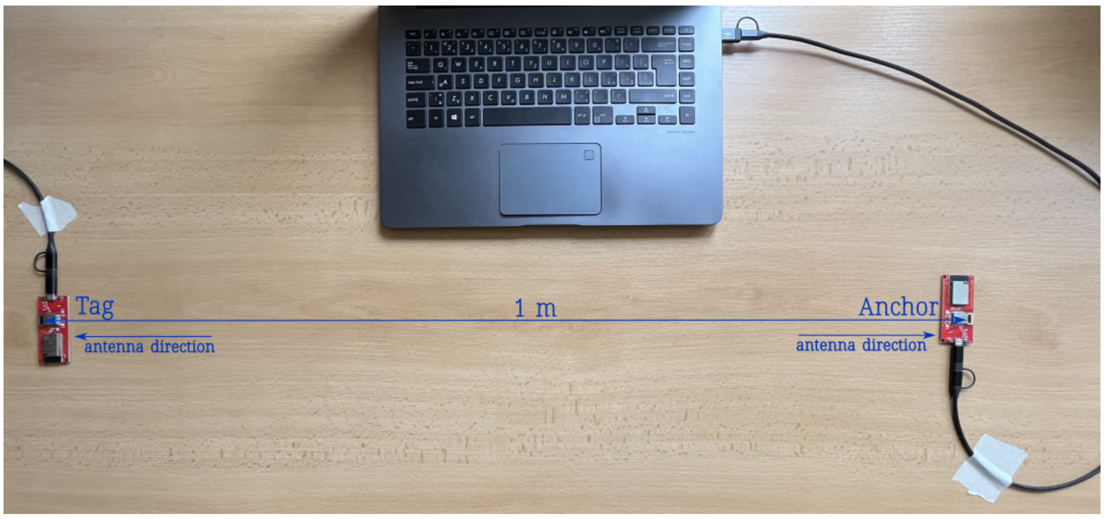


Figure 3.4 An example of the placement of the anchor and the tag, when antennas of UWB modules are directed away from each other, creating Non-line of Sight (NLoS) communication. Distance between modules is 1 meter. The anchor is connected to a laptop, enabling the observation of results logged to the serial console in the Arduino IDE. The tag is powered by a power bank.

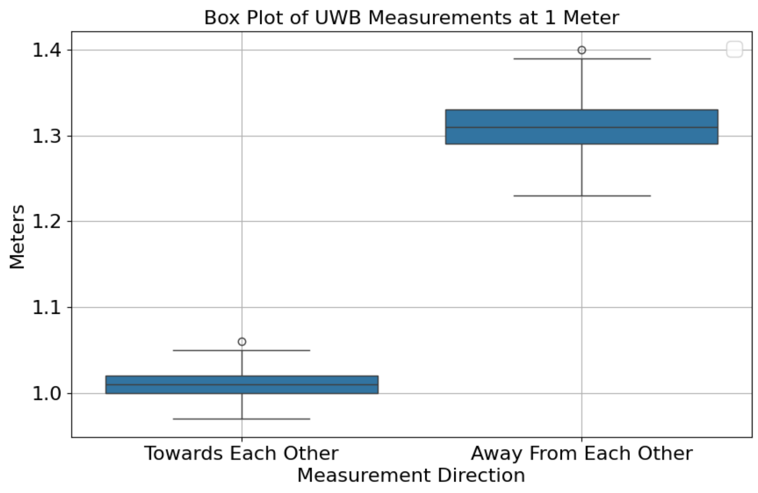


Figure 3.5 Box plot for the “Towards Each Other” and “Away From Each Other” UWB measurements at a distance of 1 meter.

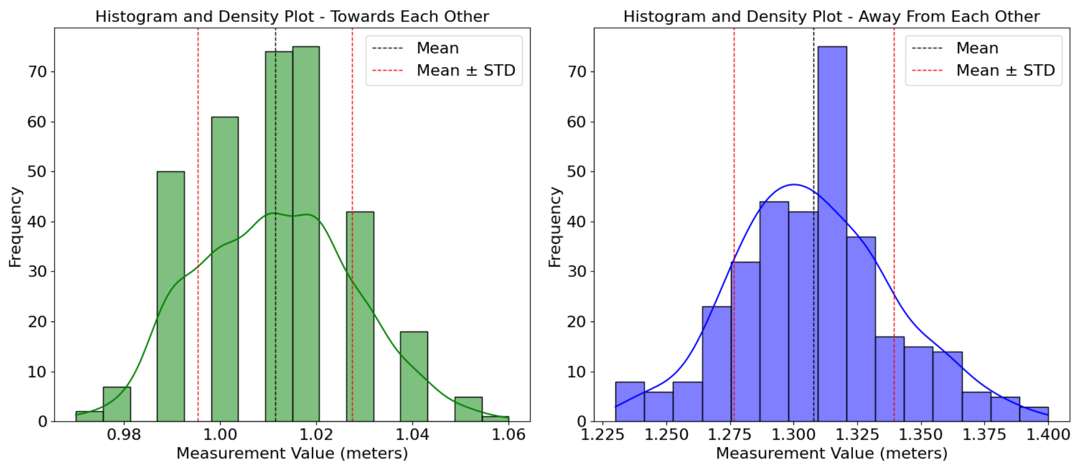


Figure 3.6 The histograms with overlaid density plots for the "Towards Each Other" and "Away From Each Other" UWB measurements. The actual distance between the tag and the anchor is 1 meter.

Statistics		
	Towards Each Other	**Away From Each Other**
Number of records	335	335
Mean (Average) (m)	1.011	1.308
Standard Deviation (m)	0.016	0.031
Minimum (m)	0.97	1.23
Maximum (m)	1.06	1.40
Variance	0.000257	0.000985

Table 3.1 Statistics for "Towards Each Other" and "Away From Each Other" measurements. Mean, standard deviation, minimum and maximum values are provided in meters.

Distance measurements	
Tag ID	Range (m)
1	0.99
1	1.02
1	1.00
1	1.04
1	1.03

Table 3.2 An example of the collected data, measured between a single anchor and a single tag. The anchor and tag are directed *"Towards Each Other"* for better accuracy of gathered data. The "From" column contains the tag address. The "Range" column lists the measured distance in meters.

3.1.3 Two Anchors – One Tag

As the next step, we attempted to scale up the system to support communication between one tag and two anchors, as depicted in Figure 3.7. This expansion is necessary for position estimation using triangulation. With all sides of the triangle and the coordinates of the anchors known, it is possible to estimate the coordinates of the tag. The anchor baseline is known beforehand.

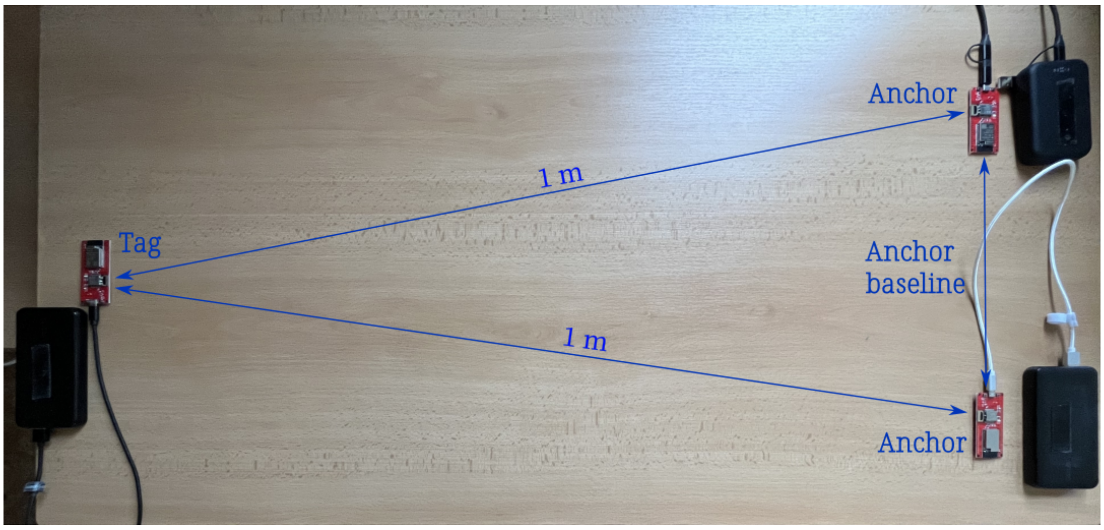


Figure 3.7 An example of the placement of two anchors and one tag. Each anchor is positioned one meter away from the tag, with both anchors are placed towards to the tag, ensuring a clear LoS for optimal communication. Each device is powered by a power bank.

However, the use of two anchors introduces the **synchronization problem** in distance measurements: to compute the tag coordinates using triangulation, distance measurements collected from both anchors need to be synchronized in time, as any time lag between measurements can result in significant errors in coordinates, especially if the tag is moving.

There are different possibilities of data synchronization. For instance, after completing the communication with a tag, an anchor can send the distance measurement including the timestamp of the measurement to a dedicated machine. Then the machine can synchronize the distance measurements received from the anchors based on the provided timestamps. Unfortunately, ESP32 UWB devices, like other radio-frequency devices, are subject to a clock drift [19]. The clock drift usually occurs when the internal clock frequency slightly deviates due to factors such as temperature variations, voltage changes, aging of the electronic components, or clock source errors. These deviations, though often small, can accumulate and lead to significant timing discrepancies over time, resulting in misaligned data measurements. Regular synchronization with a more accurate external time source, such as high-precision oscillators, can help to mitigate this issue.

A simpler method involves synchronizing the collected measurements from both anchors directly on the tag. In particular, the tag can sequentially communicate with multiple anchors, storing all the estimated distances for later position estimation.

Fortunately, there exists a solution[8] that favors the latter synchronization method, supporting a two-anchor setup with a Python-based server for data collection and storage. The big advantage of this implementation is that the server can be hosted on a separate machine, allowing to free the ESP32 board resources from an extra cost of storing the collected data.

However, a limitation of this system is that it currently supports only one operational tag in the system. The tag sequentially gathers distance measurements from the both anchors, one after another. This method effectively addresses issues with distance synchronization. After collecting the distances, the tag directly transmits this data to the server.

Additionally, the integration of the server helps to get rid of the need for the UWB devices to be connected to a laptop for data retrieval. Instead, the server can be used to receive the measurements from the tag using a wireless network.

Nevertheless, the challenge of selecting an appropriate wireless network for communication between the tag and server was the next problem we had to address. In the following section, we will delve into the various wireless network possibilities we evaluated and describe the solution we implemented to effectively address this issue.

3.2 First experiments in large environments

Before going into detail about the wireless network, let us describe our initial experiments in large environments.

Up until this point, our experiments with anchors and tags were limited to short distances, when the devices were placed on a table and the laptop served as the server receiving the measurements from the tags.

Motivated by the goal of monitoring more people simultaneously and to evaluate how precisely the UWB devices can measure distances, we decided to examine how effectively the system can perform in a more realistic conditions and whether it is possible to integrate additional tags.

3.2.1 Description of the first experiments in large environments

We performed experiments at the Faculty of Mathematics at Charles University. Figure 3.8 illustrates the `S301` corridor where these experiments were carried out. The goal of the experiments was to collect the distances estimated by the UWB devices.

It is important to note that during these experiments we considered ideal environment without any artificial obstacles that might affect the communication of UWB devices. However, we later realized that the environment naturally contains obstacles that impacted our experiments, as discussed in Section 3.4.3. Some examples of obstacles include cabinets, an electrical enclosure, and vegetation.

Additionally, we placed UWB ESP32 boards inside protective boxes to ensure their safety during our experiments. Moreover, this approach made it easier to

[8] https://github.com/Makerfabs/Makerfabs-ESP32-UWB/tree/main/example/IndoorPositioning

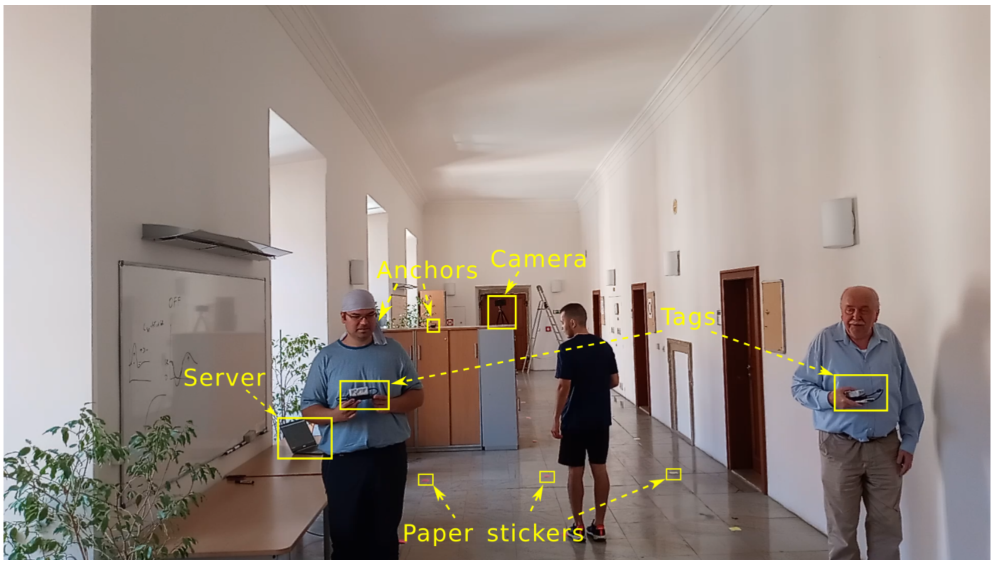


Figure 3.8 This picture serves solely as an introductory illustration of the experiment environment. It shows the `S301` experiment environment.

manage the devices throughout our experiments. An example of the secure box is shown in Figure 3.9.

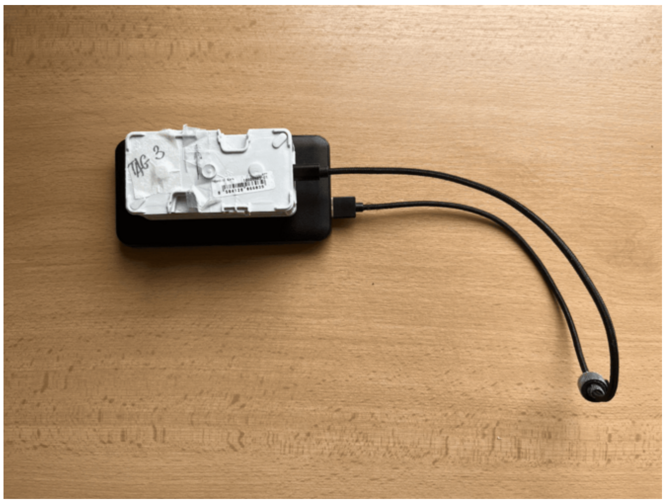


Figure 3.9 An example of a protective box that is used to protect ESP32 UWB boards.

Reference system

To effectively evaluate a distance estimation method and consequently coordinates, a reference system that provides accurate measurements is essential. Such a system enables precise evaluations of accuracy of the method we have developed.

To establish such reference system, we utilized paper stickers as markers. We placed them on the floor to outline the reference system, as shown in Figure 3.10.

Initially, the stickers were placed in three rows, maintaining an interval of 1.25 meters between each row and a 2 meter space between the stickers within a row. In subsequent experiments, we adjusted the spacing between the stickers

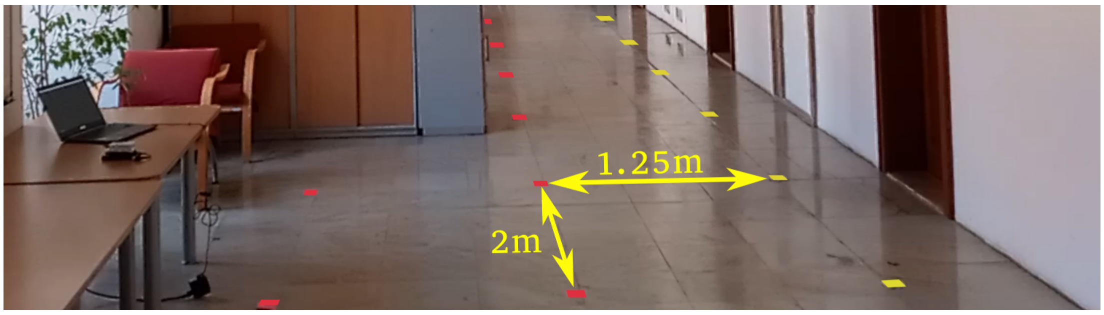


Figure 3.10 An example of the reference system used during our experiments. The stickers are placed in three rows, maintaining an interval of 1.25 meters between each row and a 2 meter space between the stickers within a row.

within each row to 1 meter and then further to 0.5 meters to enhance the precision during measurements. Throughout the rest of this work we will use $\mathtt{RS}_{\mathtt{1m}}$ and $\mathtt{RS}_{\mathtt{0.5m}}$ notations respectively to indicate a specific reference system.

During the experiments, we held the tags in our hands and walked through the experiment environment, stopping at each paper sticker for a certain period of time. This allowed us to collect data corresponding to each specific sticker, for which we knew the actual distance from the baseline and its coordinates. We used this information to evaluate the measured distances. The analysis of the collected data is provided in Chapter 7.

3.2.2 The data collected during experiments

An example of $\mathtt{E}_5\mathtt{(DA\_S301\_S}_6\mathtt{(T3\_A2\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{p}\mathtt{))}$ experiment conducted in S301 environment is shown in Figure 3.11 (the notation is explained below).

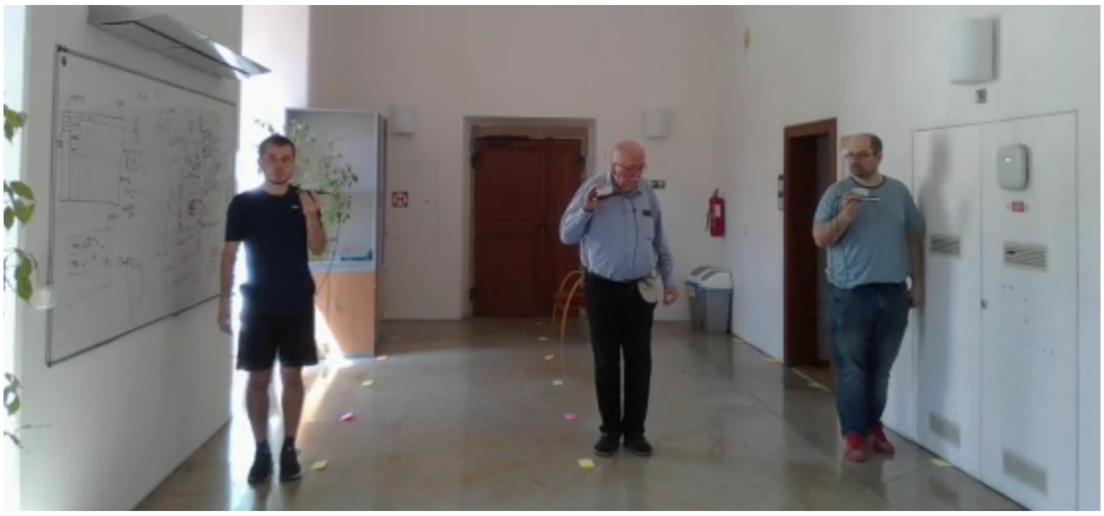


Figure 3.11 An example of the experiment $\mathtt{E}_5\mathtt{(DA\_S301\_S}_6\mathtt{(T3\_A2\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{p}\mathtt{))}$

Throughout the rest of this work, we will use the **specific notation**, which will simplify the experiment description. For example, $\mathtt{E}_5\mathtt{(DA\_S301\_S}_6\mathtt{(T3\_A2\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{p}\mathtt{))}$ $\rightarrow \mathtt{D}_{\mathtt{raw}}\mathtt{(E}_5\mathtt{)}$ represents 5th experiment. The type of experiments is data acquisition DA, conducted in the S301 environment. It follows the $\mathtt{S}_6$ scenario (described in Section 7.1). In this scenario, three people synchronously walk along the paper sticker rows, stopping at each paper sticker for one second. They walk in both directions. Each person walks solely along a certain sticker row. The experiment involves the use of 3 tag and 2 anchors. The anchors are placed on the cabinet. Tags are placed on hands ($\mathtt{TP_h}$). The experiment involve dynamic movement ($\mathtt{M_d}$) on predefined lines ($\mathtt{W_p}$) only. The set of the collected **raw** distances is then represented as $\mathtt{D_{raw}(E_5)}$. Please refer to Appendix A for more detail about this notation.

An example of collected $\mathtt{D_{raw}(E_5)}$ set of distances is shown in Table 3.3. It consists of the measurement timestamp, the Tag ID, the Anchor ID, and the corresponding distance to that anchor.

Distance measurements					
Timestamps	Tag ID	Anchor ID	Distance	Anchor ID	Distance
1685796766737	1	101	4.81	102	5.07
1685796767011	2	101	5.15	102	5.05
1685796767286	3	101	5.95	102	5.54
1685796767558	1	101	4.79	102	5.04
1685796767831	2	101	5.08	102	4.94
1685796768103	3	101	5.87	102	5.43
1685796768376	1	101	4.77	102	5.01
1685796768648	2	101	4.87	102	4.74
1685796768918	3	101	5.73	102	5.25

Table 3.3 An example of the data $\mathtt{D_{raw}(E_5)}$ collected during $\mathtt{E_5}$.

Together with the collection of the UWB data, we recorded a video of the experiment. Throughout the rest of this work, we will use the `VD(expid)` notation to represent video data, where `expid` is the identifier of the experiment. An example of the $\mathtt{VD(E_5)}$ video frame is shown in Figure 3.11.

Throughout the work, we have conducted lots of different experiments. The data collected during each experiment is available in main repository of this thesis on GitHub[9]. The data corresponding to the $\mathtt{E_5}$ experiment is available [10].

First attempts in synchronization of video and UWB data

When synchronizing data between different sources, it is important to ensure accurate data alignment. The data can either be collected simultaneously and synchronized immediately, eliminating the need for later synchronization, or collected separately, requiring subsequent synchronization. The former approach makes the both data monolithic, which are later difficult to split and process, while latter allows each dataset to be used independently, but requires precise timestamps for accurate data alignment.

We considered the second approach, as our goal is to collect data that can be used by others to develop new localization methods, or test existing.

[9]https://github.com/Razyapoo/Master-Thesis/tree/main/Recorded Experiments

[10]https://github.com/Razyapoo/Master-Thesis/tree/main/Recorded Experiments/Experiments (4-6) 2023.03.06/Experiment 5

Initially, we recorded UWB data along with the timestamps and used a mobile phone to record videos. Figure 3.8 shows an example view from a mobile phone camera. We utilized an acoustic signal to indicate the start of the experiment. This sound served as an indicator from which the data was considered synchronized. However, this method introduced several complexities. For example, a video provided relative timestamps rather than absolute timestamps, making it difficult to align the data precisely.

Subsequently, we switched to using **webcams** for video recording. This approach is similar to connecting to CCTV cameras and allows recording **absolute timestamps**. We synchronized the data by minimizing the absolute difference between frame timestamps:

$$min|TS_{video}(i) - TS_{UWB}(j)|$$

where $TS_{video}(i)$ represents the timestamp of the i-th video frame and $TS_{UWB}(j)$ represents the timestamp of the j-th UWB data.

We have adopted this method in our subsequent experiments, with further details available in Chapter 4.

3.2.3 Communication with the server

During our first experiments, the server was operating in a passive listening mode, where it only received and processed distance data sent by a tag. This formed a decentralized architecture of the system, where a tag independently initiated a connection with anchors, calculated the distance from them and sent the measurement result to the server.

Public access points

The initial approach considered for establishing communication between the server and tags was the use of public access points and wireless networks, which are usually pre-installed in environments like hospitals or industrial premises. This seemed advantageous due to an existing infrastructure, which could potentially simplify deployment.

The Faculty of Mathematics at Charles University has its own MQ Telemetry Transport (MQTT) server utilized for a communication of IoT devices. After trying to use it, we encountered significant challenges in transmitting data through the public network and subsequently accessing this data on the server, primarily due to the privacy restrictions.

It is also important to note that utilizing a public network can potentially affect the communication of other components within network. Therefore, we decided to establish our own private network via a laptop, which fortunately resolved the connectivity issue. In addition, this network is only needed during the deployment stage for the communication with the ESP32 UWB devices.

3.2.4 Inconsistency in the data

As the next step, we tried to involve two tags to communicate with anchors. We assumed that the system was functioning correctly and data was accurately measured for both tags. However, after several experiments, either by accident or fortunately, we ran into a problem.

Occasionally, the distance measurements collected and related to one tag actually contained measurements related to another tag. This was due to the implementation of the Two-Way Ranging (TWR) protocol provided by the original DW1000 library, which did not function as expected. Specifically, with **two tags and one anchor**, the tag positioned closer to the anchor **overwrote** data intended for the tag positioned farther away.

Please refer to Figure 3.12, which illustrates the anchor positioned 1 meter from Tag 1 and 0.5 meters from Tag 2. The anchor and tags are aligned on the same line. UWB antennas of the both tags are directed towards the anchor's antenna ensuring a clear LoS between each tag and the anchor.

It is important to note that the **inconsistencies** in the data were observed in all cases where one tag was positioned closer to the anchor than another tag, regardless of whether all three devices were aligned on the same line or not. However, the alignment illustrated in Figure 3.12 not only leads to data overwriting, but also causes conflicts between tags, resulting in signal interruption and attenuation, and preventing tags from communicating with the anchor. This is a very important observation, as it helped us with our further experiments.

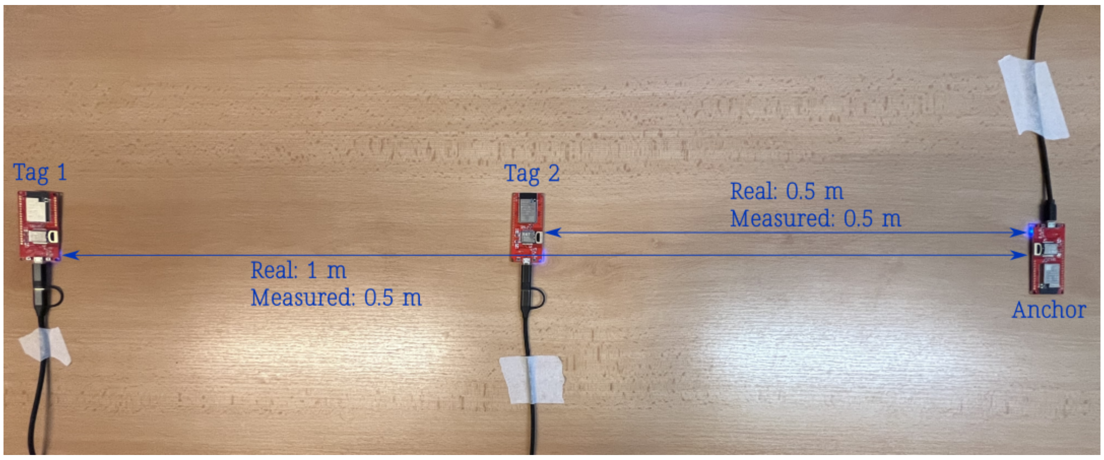


Figure 3.12 An example of the setup leading to **inconsistency** in data collected by the tags. The anchor and the tags are aligned on the same line. Each device is powered by a power bank (not visible on the image). This alignment not only leads to data overwriting, but also causes conflicts between tags, resulting in signal interruption and attenuation, and preventing tags from communicating with the anchor.

This issue arose because the original implementation stores the calculated distance measurements between the anchor and tags in a shared global variable on the anchor side. This variable is not specifically allocated for a particular tag, which leads to inconsistencies in the data measured using tags.

A possible solution to this problem would be to have the anchors directly send the calculated distances to the server. However, ensuring that the distance measurements collected from different anchors are properly synchronized and correctly associated with a specific tag presents a significant challenge, as was discussed in Section 3.1.3.

We decided not to rewrite the entire library, because we were unclear about its details, and changing it could lead to new possible issues that might take a long time to resolve.

Intuitively, it was clear that once a tag establishes communication with an anchor, that anchor should be exclusively reserved for that tag and not accept messages from other tags. Furthermore, after the distance is calculated, the anchor should complete the communication with the current tag by sending the distance to it before switching to an another tag. Instead, in the original library, the tag requests the measured distance from the anchor after the communication has completed. This setup poses a risk since the measured distance, stored in the shared variable, can be overwritten by another tag at any time, as happened to us during our experiments.

Fortunately, we discovered a code on GitHub[11] that is written by Seokseong Jeon. This code seemed promising to us for working with a larger number of tags. It implements the DS-TWR protocol in the similar way as the official DW1000 library. In contrast, it includes examination of source and destination addresses and utilizes different states to manage the communication between the anchors and tags.

However, we have encountered difficulties in enabling communication between the anchors and the tags, as they were not communicating at all. The reason is that the provided system is designed specifically to work with the standalone DWM1000 module[12], which utilizes the Arduino Pro Mini to enable the interaction with the Raspberry Pi, which transfers the collected data over the network to the server.

In contrast, our setup uses UWB ESP32 boards from Makerfabs. They integrate Bluetooth and WiFi capabilities, enabling them to communicate with the server over the wireless network directly without requiring any additional hardware.

As a result, we reimplemented the system to ensure proper communication and functionality.

3.3 The implementation of our own IoT network

Before going into the details of code changes, it is essential to understand the Two-Way Ranging (TWR) protocol.

3.3.1 An overview of existing ranging protocols

The distance between anchors and tags can be measured using a ranging algorithm, which calculates Time of Flight (ToF) of the signal, such as radio wave or light. This measurement can be calculated using different techniques, including One-Way Ranging (OWR), Two-Way Ranging (TWR), and Time Difference of Arrival (TDoA)[13].

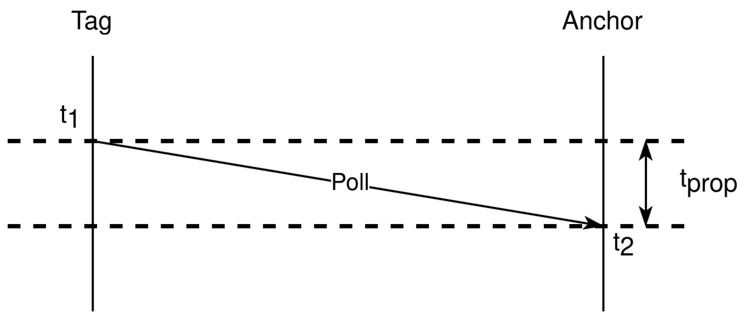


Figure 3.13 One-Way Ranging [20].

One-way Ranging

The One-way Ranging (**OWR**) involves a single transmission from a sender to a receiver. Please refer to Figure 3.13. *Tag* transmits the "Poll" message, including the time t_1, representing the transmission time. Upon receiving this message, *Anchor* records the reception time and calculates the propagation time t_{prop}, which represents ToF. The main disadvantage of the OWR is that it requires both the devices to have synchronized internal clocks [21], which may lead to the clock drift. ToF is then equal to t_{prop}.

Two-Way Ranging

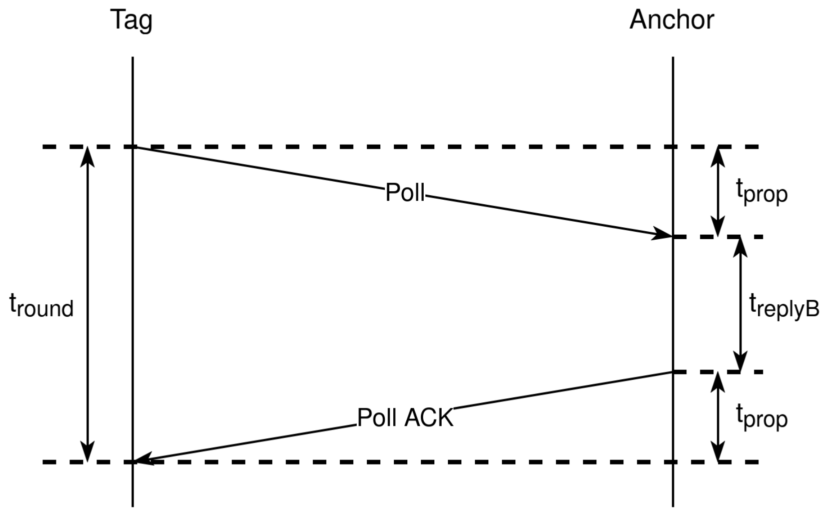


Figure 3.14 Two-Way Ranging [20].

Two-Way Ranging (**TWR**), on the other hand, requires both the devices to exchange messages. The distance is calculated based on the time it takes for a signal to travel from one device to another and back. Please refer to Figure 3.14. A *Tag* initiates the exchange by transmitting a "Poll" message to an *Anchor*, which then logs the reception time of the message. Subsequently, the *Anchor* waits for a certain period, t_{replyB}, and sends back a "Poll ACK" message containing the

[11]https://github.com/somidad/dw1000-positioning/
[12]https://www.qorvo.com/products/p/DWM1000
[13]https://en.wikipedia.org/wiki/Time_of_arrival

transmission time and the time when “Poll” message was received, **based on its own clock**. Upon receiving the “Poll ACK” message, the *Tag* logs the time of the reception, **based on its own clock**, and proceeds to calculate the round-trip time t_{round}.

This method has an advantage of being asynchronous, as it eliminates the strict synchronization between anchor and tag [21]. The Time of Flight (ToF), or t_{prop}, is then computed by the following formula:

$$t_{\text{prop}} = \frac{1}{2}(t_{\text{round}} - t_{\text{replyB}})$$

where t_{round} denotes the round-trip time, which is stored in *Tag*, and t_{replyB} denotes the time delay for *Anchor*, after which it sends the response signal to *Tag*.

However, UWB devices typically have **clock drift** issues, which leads to significant errors in the ToF estimates. These errors increase as t_{replyB} increases [22].

Double-Sided Two-Way Ranging

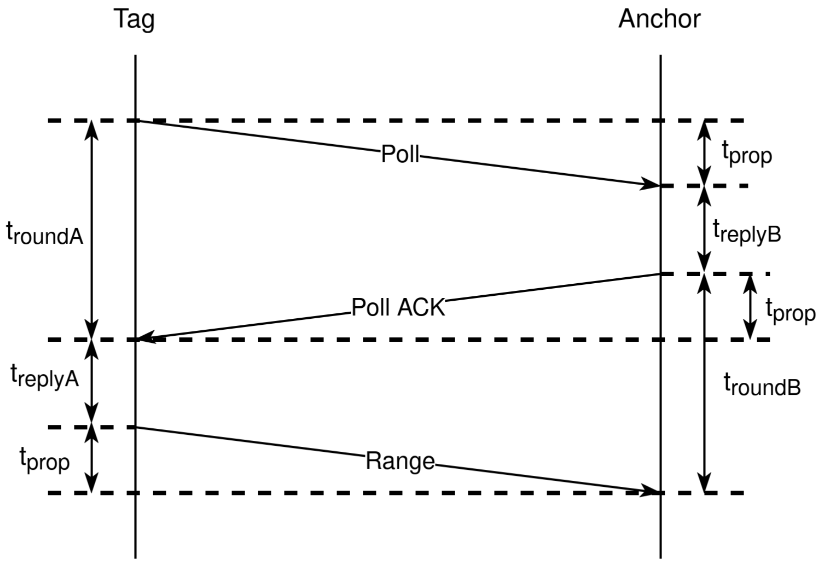


Figure 3.15 Double-Sided Two-Way Ranging [20][23].

Double-Sided Two-Way Ranging (**DS-TWR**) protocol involves two round-trip exchanges to compute the distance between two devices. See figure Figure 3.15. *Tag* initiates the first round-trip exchange by sending “Poll” message, to which *Anchor* responds by sending “Poll ACK” message, thus initiating the second round-trip exchange. *Tag* then responds to *Anchor* by the “Range” message, completing the full DS-TWR exchange. Both devices record the transmission and reception times of the messages, allowing for precise calculation of the distance between them.

There are two types of DS-TWR protocol: *symmetric*[14] and *asymmetric*. The symmetric version requires the reply times from each device to be the same.

[14]https://en.wikipedia.org/wiki/Symmetrical_double-sided_two-way_ranging

However, this can cause the **clock drift**, significantly affecting the accuracy of the ToF measurement [22].

To reduce this error, Neirynck, Luk and McLaughlin proposed an asymmetric version of TWR protocol [22], which does not require same reply times for each device. The resulting ToF estimate, t_{prop}, is then calculated using the following expression:

$$t_{\text{prop}} = \frac{t_{\text{roundA}} \cdot t_{\text{roundB}} - t_{\text{replyA}} \cdot t_{\text{replyB}}}{t_{\text{roundA}} + t_{\text{roundB}} + t_{\text{replyA}} + t_{\text{replyB}}} \tag{3.1}$$

In this work, we implement the asymmetric version of the Double-Sided Two-Way Ranging (DS-TWR) protocol and use Equation (3.1) to compute ToF of signals.

Distance calculation

Since UWB utilizes radio waves for communication, the distance between the *Anchor* and the *Tag* can be calculated using the following formula:

$$d = c \cdot t_{\text{prop}} \tag{3.2}$$

where c is the speed of light, t_{prop} is the estimated ToF, and d is the actual distance between the *Anchor* and the *Tag*.

3.3.2 Discovery and Ranging phases

As illustrated in Figure 3.16, there are two phases in which both the anchor and the tag operate: *discovery* and *ranging*. In the discovery phase, the tag discovers the desired number of anchors available in the current network. In the ranging phase, the tag iteratively performs the Double-Sided Two-Way Ranging (DS-TWR) protocol with each discovered anchor to measure the distance to it.

3.3.3 Decentralized architecture

In this section, we will describe our first approach, which is aimed to ensure the correct operation of UWB devices, and allows us to use more tags simultaneously. It is presented for solely for informative purposes, to show what challenges we have encountered and why we decided to switch to centralized architecture. It also details the DS-TWR protocol for its better understanding.

In the next section, Section 3.3.4, we will describe our **final solution**, which we implemented from scratch. It has another implementation details, however the basic idea of the DS-TWR protocol remains the same.

The code provided by Seokseong Jeon on GitHub[15] involves different states in communication between the anchors and the tags. **We revised** these states **to better meet our requirements** and removed unnecessary components responsible for interactions with the Arduino Pro Mini and Raspberry Pi. In the following sections, we will describe the initialization and communication process of anchors and tags.

[15]https://github.com/somidad/dw1000-positioning/

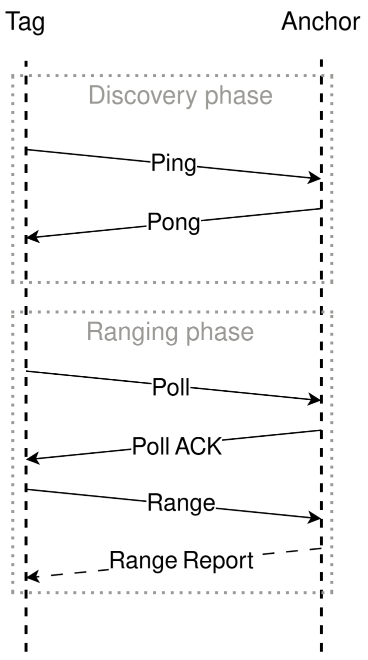


Figure 3.16 Discovery and Ranging phases during message exchanges [23].

Device initialization

Before the actual communication can start, both the anchors and the tags need to be activated.

First, the anchors are activated and initialized with necessary settings. After completing the setup, they transition to the listening mode, `STATE_IDLE`, and wait for incoming requests from the tags, as shown in Figure 3.17.

Next, the tags are switched on. They are activated sequentially to avoid signal attenuation and interruptions, which occur due to the fact that all tags start sending “Blink” request messages to the anchors at the same time. As a result an anchor is unable to process any incoming requests from tags. This is the issue we addressed during our first experiments. Upon initiation, the tags connect to the server over the private wireless network, and transition to the `STATE_IDLE`. The server starts passively listening for incoming messages from the connected tags.

We assigned each UWB device a unique identifier based on its MAC address, which facilitated their management and identification. To ensure clear differentiation between devices, we allocated mutually exclusive ranges of addresses for the anchors and the tags, such that the anchors process only messages from tags, and tags process only messages from anchors.

The tag coordinates are calculated using triangulation, which requires distance measurements from at least two anchors. However, these measurements need to be synchronized in time for precise estimation of the coordinates. As discussed in the Section 3.1.3, there are two different ways to synchronize data, including synchronization on the server or synchronization directly on the tag. We chose the latter, because it ensures that synchronization is handled automatically on the tag, as distances are collected sequentially one by one. This avoids the potential clock drift issues associated with timestamp-based synchronization on the server.

Discovery phase

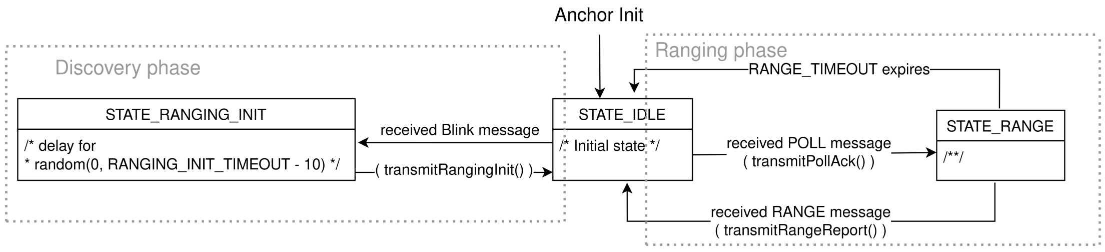
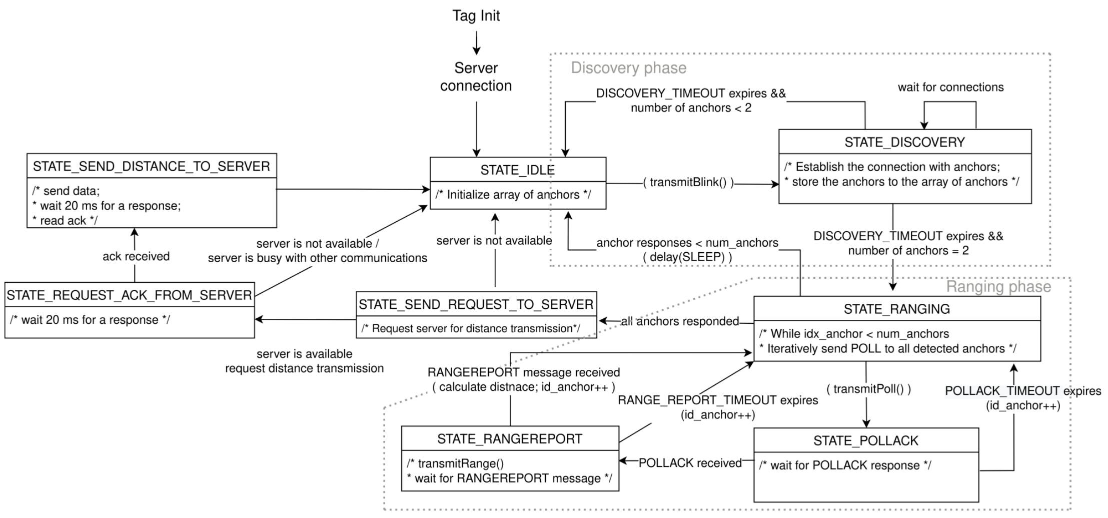


Figure 3.17 Decentralized architecture. **Updated states** used in an anchor during the communication between an anchor and a tag [24].



Tag Init
Server connection
Discovery phase
DISCOVERY_TIMEOUT expires && number of anchors < 2
wait for connections
STATE_DISCOVERY
/* Establish the connection with anchors;
* store the anchors to the array of anchors */
STATE_SEND_DISTANCE_TO_SERVER
/* send data;
* wait 20 ms for a response;
* read ack */
STATE_IDLE
/* Initialize array of anchors */
(transmitBlink())
ack received
server is not available / server is busy with other communications
server is not available
anchor responses < num_anchors (delay(SLEEP))
DISCOVERY_TIMEOUT expires && number of anchors = 2
Ranging phase
STATE_REQUEST_ACK_FROM_SERVER
/* wait 20 ms for a response */
STATE_SEND_REQUEST_TO_SERVER
/* Request server for distance transmission*/
all anchors responded
STATE_RANGING
/* While idx_anchor < num_anchors
* Iteratively send POLL to all detected anchors */
server is available
request distance transmission
RANGEREPORT message received
(calculate distnace; id_anchor++)
RANGE_REPORT_TIMEOUT expires
(id_anchor++)
(transmitPoll())
POLLACK_TIMEOUT expires
(id_anchor++)
STATE_RANGEREPORT
/* transmitRange()
* wait for RANGEREPORT message */
POLLACK received
STATE_POLLACK
/* wait for POLLACK response */

Figure 3.18 Decentralized architecture. **Updated states** used in a tag during the communication between an anchor and a tag [24].

To start the **discovery phase** the tag transmits a “Blink” signal aimed at discovering all the anchors available in the current UWB network, and transitions to a STATE_DISCOVERY waiting for responses from the anchors.

Upon receiving the “Blink” signal and checking the source address of the tag, the anchor transitions to a STATE_RANGING_INIT and waits for a random delay, which is necessary to prevent the tag from being overloaded with responses from the anchors. This is due to the fact, that an UWB device, as any other radio-wave device, requires a time to process the incoming signals [23]. After the random delay expires, the anchor sends a “Ranging Init” signal back to the tag, indicating its presence in the UWB network and then transitions back to the STATE_IDLE.

Upon receiving the “Ranging Init” signal, the tag examines the destination address contained in the message. If the message is intended for it, the tag remembers the identifier of the discovered anchor for later communication. The tag only waits for incoming “Ranging Init” signals from anchors for a certain period of time, DISCOVERY_TIMEOUT. If the desired number of anchors has not been discovered after that timeout, the tag returns to the STATE_IDLE to initiate a new discovery phase. Otherwise, the tag transitions to a STATE_RANGING, thereby initiating a **ranging phase**.

Ranging phase

Starting from this point and throughout the DS-TWR communication process, both the anchor and tag consistently check addresses contained in the message. If either the source or destination address does not match, they ignore that message.

During the `STATE_RANGING` the tag iteratively perform the DS-TWR protocol with each discovered anchor by transmitting a "Poll" message and recording the transmission time. It then transitions to a `STATE_POLLACK` and waits for the response from the anchor for `POLLACK_TIMEOUT` milliseconds. If no response is received within the defined timeout, the tag returns back to the `STATE_RANGING` to initiate the communication with an another anchor.

Upon receiving the "Poll" message, the anchor records the reception time, waits for a predefined t_{replyB} delay, and sends the "Poll ACK" message back to the tag. Then it transitions to a `STATE_RANGE` and waits for `RANGE_TIMEOUT` milliseconds for the response.

Upon receiving the "Poll ACK" message, the tag records the reception time, waits for a predefined t_{replyA} delay, sends a "Range" message back to the anchor and records the message transmission time. Then it transitions to a `STATE_RANGEREPORT` and waits for `RANGE_REPORT_TIMEOUT` milliseconds for the response from the anchor. If there is no response, the tag transitions back to the `STATE_RANGING` to initialize the communication with an another anchor.

In order to solve the problem discussed at the beginning of Section 3.2.4 and depicted in Figure 3.12, where data collected and related to one tag contained measurements related to another tag, we decided to compute the actual distance on the tag side.

Therefore, if the anchor successfully receives the "Range" message, it records the message reception time, sends all the collected reception timestamps in the response message and transitions to the `STATE_IDLE`, thereby concluding the communication process from its side.

Upon receiving the "Range report" message, the tag calculates the ToF. It then determines the actual distance from the anchor using Equation (3.2).

Server implementation in C++

Up to this point, we have been using a server provided by Makerfabs, which allowed us to work with only one tag. We updated it to support two tags by assigning each tag a specific port.

As we were trying to support more tags, it became impractical to manually add a new port for each newly connected tag.

Eventually, we decided to implement our own server in C++, as it gives more control over server management.

Server initialization

The server has two responsibilities. The first responsibility is to handle new connection requests from the tags. Another responsibility of the server is to handle incoming requests containing estimated distances from the tags that are already connected to the server.

Upon completing the distance estimation, the tag transitions back to the STATE_RANGING. In this state, if the desired number of distance measurements has been collected, the tag sends them to the server. Otherwise, it returns to the STATE_SCAN and initiates the discovery phase again.

Adding states to the server

An issue occurred when a tag transitioned directly to STATE_SCAN without delay, immediately starting a new communication with anchors and preventing other tags from interacting with these anchors. Additionally, if a tag failed to collect enough distance measurements due to communication issues with a particular anchor, it would get stuck in a cycle without progressing. To address this, we introduced a delay, requiring the tag to wait after sending the measurements to the server and before starting a new discovery phase.

Furthermore, even when a tag was already connected to the server, it was crucial to first notify the server about the intention of the tag to send the distance measurements before the actual submission. Therefore, as illustrated in Figure 3.19, we introduced two different server states: STATE_IDLE and STATE_COMMUNICATION. Additionally, as illustrated in Figure 3.18, we added three more states to a tag: STATE_SEND_REQUEST_TO_SERVER, STATE_REQUEST_ACK_FROM_SERVER, and STATE_SEND_DISTANCE_TO_SERVER.

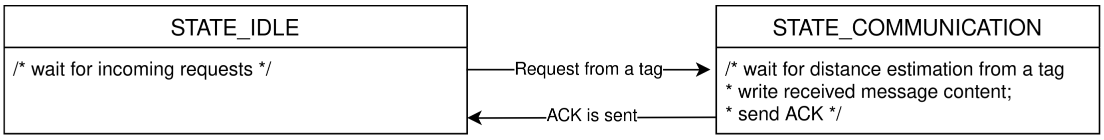

STATE_IDLE
/* wait for incoming requests */
Request from a tag
ACK is sent
STATE_COMMUNICATION
/* wait for distance estimation from a tag
* write received message content;
* send ACK */

Figure 3.19 Decentralized architecture. Server states.

To initiate various actions during the communication, the server and the tags use different request flags. For instance, “REQUEST” signals to the server that the tag is ready to transmit measured distances to the server.

Therefore, the server initially remains in the STATE_IDLE, waiting for incoming requests from a tag. If a tag is in the STATE_RANGING and has collected enough distance estimates, it transitions to the STATE_SEND_REQUEST_TO_SERVER and verify if the connection with the server is still active. If the connection has been lost, the tag transitions back to the STATE_IDLE and begins a new discovery phase. Otherwise, the tag sends a request flag “TRANSMIT” to the server to initiate the distance transmission process. It then transitions to the STATE_REQUEST_ACK_FROM_SERVER and waits for a response from the server. If there is no response within a certain period, for example if the server is busy communicating with another tag, the tag transitions to the STATE_SCAN and initializes a new discovery phase. We fixed this gap in the implementation of centralized architecture by retrying the submission again.

Upon receiving the request from the tag, if the server is available, it records the tag identifier and sends an acknowledgement flag “ACCEPTED” to indicate successful receipt of the request. Then it transitions to the STATE_COMMUNICATION, as illustrated in Algorithm 1. This state is essential, as it shows that the server is currently involved in communication with the tag. This prevents other tags from interrupting the server while it is already communicating with the particular tag.

Algorithm 1 Getting a request from a tag

```
if state = STATE_IDLE and FLAG(buffer, ”REQUEST”) then
    tagId ← EXTRACTTAGID(buffer)
    SENDTOTAG(tag, ”ACCEPTED”)
    state ← STATE_COMMUNICATION
end if
```

After receiving the acknowledgment signal, the tag transitions to the `STATE_SEND_DISTANCE_TO_SERVER` state, sends the measured distances and its identifier to the server, and waits for a predefined period of time for confirmation. This specific time period is needed to transmit the messages back and forth between the tag and the server. This timeout period should ensure that the tag has enough time to receive a response from the server. If the period is too short, the tag will not be able to receive a response from the server.

Upon receiving the measured distances, the server verifies the source. If the request comes from an unexpected tag, the server ignores it and waits for the expected one. Once the correct request is received, the server stores the measurements, sends the acknowledgement flag “RECEIEVED” to the tag, indicating successful receipt, and returns to the `STATE_IDLE`, where it resumes listening for new requests from other tags. Refer to the Algorithm 2 for more details.

Algorithm 2 Receiving the distance measurements on the server

```
if state = STATE_COMMUNICATION and MATCH(buffer, tagId) then
    receivedData ← EXTRACTDATA(buffer)
    SAVEDATA(receivedData)
    SENDTOTAG(tag, ”RECEIVED”)
    state ← STATE_IDLE
    CLEARTAGID(tagId)
end if
```

The `STATE_SEND_DISTANCE_TO_SERVER` is specifically designed for transmitting the distance measurements and for receiving acknowledgement from the server. The tag’s communication states were designed for possible future expansions. For instance, if the tag does not receive a response from the server within a certain time, it will reattempt to send the measured distance instead of returning to the discovery phase. However, this reattempt should occur only once to prevent repeated signal transmissions that could cause delays or even block incoming messages from other tags. We added this possibility in the centralized version of the UWB system.

Weaknesses of the decentralized system

Eventually, server states helped eliminate server overload from direct data transmissions. Previously, when multiple tags sent messages simultaneously, the server often failed to separate and correctly process these messages, resulting in mixed, lost, or delayed data.

While the state integration resolved many issues with multiple tags, **challenges remained**. We were only able to gather data on the server from a maximum of two tags at once. In addition, the data did not arrive at the server as frequently as was expected because the tags were competing to communicate with the anchors. Simultaneous requests from tags often resulted in signal collisions, preventing anchors from responding.

To address this problem, we modified the condition requiring a tag to collect data from at least two anchors before sending it to the server. We reduced the number of required anchors to one. Additionally, this adjustment allowed us to add a third tag to the system. However, we began receiving many messages on the server that only contained measurements from one anchor, despite having two or three anchors active. As a result, the entire system began to operate slower because received distance measurements were from only one anchor, and the overall frequency of data related to a single tag was lower.

We have often encountered situations where **anchors became non responsive** without any apparent reason. They did not receive any requests from the tag. This could happen if the number of interrupts occurring in a short period exceeded the processor ability to handle them efficiently, or if interruption flags were not cleared properly after handling an interrupt. As a temporary solution, we implemented a timeout mechanism to detect when an anchor gets stuck. After timeout, the anchor performs a hard reset and re-initializes with its initial settings. While this is not the ideal solution, it helped to mitigate the issue at the cost of the delay it takes to reset an anchor.

During the ranging phase, as soon as the anchor and the tag established a connection, they recorded the address of their counterpart to ensure that future messages would arrive from the correct source. However, this led to a situation where an anchor, upon receiving numerous requests from other tags, spent time checking their addresses and informing them that it is already busy. As a result, the anchor could not respond to the tag it is currently communicating with. This delay often caused timeouts designed to monitor if the counterpart was alive and responding. Consequently, it led to communication breakdowns between the anchor and the tag.

An important issue occured when three tags operated simultaneously with two anchors. The server often received data related only to one particular tag. This tag attracted all the attention of the anchors. **We addressed this issue by** implementing the list of recently communicating tags. In particular, when a tag estimates the distance to a certain anchor, that anchor adds that tag to its list of recently communicating tags. Then upon receiving a new request from a tag, the anchor starts communication only if the tag is not already in its list. The size of this list can vary. In our experiments, we utilized a list of size one.

However, the use of least recently communicating tags did not significantly improve the situation, since communication was often disrupted. This issue arose because a tag might be in the list of one anchor but not in the list of another, and vice-versa for another tag. Such discrepancies in the lists between different anchors led to inconsistent communications. This situation led us to consider the potential benefits of developing a centralized version of the system.

To address all these problems, we explored several potential solutions. One promising approach involves assigning each tag its own dedicated frequency channel for interactions with the anchors. Throughout the communication, an anchor tunes to the same channel as the active tag. After completing the interaction, the anchor switches to the channel of an another tag. However, while this method eliminates the issue of the signal collision and competition among tags, it is time-consuming due to the need for anchors to continually switch channels [25].

Through our experience with UWB devices, **we have begun to understand** why tags might interrupt each other. As mentioned in Section 3.3.1, during the DS-TWR communication process, both the anchor and the tag await reply delay upon receiving the message from the counterpart. This delay requires careful adjustment because it can lead to mutual interruptions between devices if they transmit signals simultaneously. This problem applies especially to tags, as synchronization between them is hard to achieve. While the use of different channels might help, **another solution** is to implement a **centralized system** where the server acts as an arbitrator in communications between anchors and tags. In such systems, the synchronization needs to be managed only between the anchors and only during the ranging phase.

3.3.4 Centralized architecture - Our final solution

In this section we describe **our own implementation** of the DS-TWR protocol, which finally enabled us to work with three tags simultaneously.

In the centralized architecture, the server acts as an arbitrator, controlling the communication process. It determines the sequential order in which tags should communicate with anchors, ensuring that only one tag communicates with the anchors at a time.

The centralized architecture has many benefits. For instance, it reduces the load on the server and eliminates competition between tags for the attention of the anchors. It reduces the time, which an anchor spends checking the source address and handling messages from multiple tags. This architecture helps reduce the complexity of synchronizing delays during the DS-TWR communication process. Finally, its **key benefit** is the ability to control the order in which the tags communicate with the anchors, allowing for enhanced management of the communication process. All these benefits were confirmed during the testing phase.

However, there are some **drawbacks**. For example, the frequency of data reception is lower than in a decentralized system, because each anchor must passively wait after communicating with one tag before it can proceed with others. Eventually, in the centralized version, we achieved even higher frquency of the received data compared to the decentralized version.

After considering all the advantages and disadvantages, we decided to implement our own UWB network based on a centralized architecture.

Our adaptation of the server for centralized architecture

In the decentralized architecture the server acts primarily as a passive listener, waiting for incoming requests from tags. It cannot send outgoing requests to instruct a tag to start interacting with anchors. To address this limitation we have explored several approaches.

One such approach involves the tag frequently asking the server to check its availability for communication. If the server is available, it responds to the tag with a command to begin communicating with anchors. This command can then be interpreted as “Measure!”. If the server is not available, it simply ignores the incoming message from a tag.

Upon receiving the “Measure!” command form the server, the tag initiates communicating with the anchors. This approach ensures that only one tag communicates with the anchors at a time, preventing the anchor from being overloaded by multiple simultaneous requests from the tags.

However, a downside of this method is that it can lead to the server being overloaded by continuous requests from the tags, requesting the permission to start their communication with the anchors. This affects an ability of the server to effectively manage communications and respond in time to the tags.

This issue highlighted the need for a more reliable server management method that can effectively handle high volume of incoming requests without the delays in responses.

Considering this potential problems, **we have introduced a queue of connections**, which is managed by the server, and **eliminated all the server states to simplify the architecture.**

The server operates in two main modes. In the first mode, it accepts new connections from tags. In particular, upon activation, each tag sends a request to the server, indicating its presence on the network. After receiving the request, the server adds a tag to the queue of connections.

In the second mode, if the server is currently not busy and the queue is not empty, the server selects a tag from the queue and sends it the “Measure!” request flag to initialize communication with anchors, as illustrated in Algorithm 3. This code is available in the main repository of the thesis on GitHub[16]

Algorithm 3 Server’s request to initiate communication with anchors.

```
1: if !tagQueue.empty() and !isServerBusy then
2:     tag $\leftarrow$ tagQueue.pop()            ▷ Select next tag for communication
3:     SENDTOTAG(tag, ”Measure!”)
4:     isServerBusy $\leftarrow$ true
5: end if
```

As in the decentralized architecture, both the server and tags in **the centralized system use request flags** to initiate various actions. To simplify the communication process, we reduced the number of signals to two. Specifically, “Measure!” is used to signal a tag to start communication with the anchors, and “RECEIVED” is used to confirm the successful receipt of distance measurements by the server. Furthermore, we introduced `isServerBusy` indicator, which shows when the server is engaged in active communication with a specific tag. This prevents the server from sending a new request to another tag from the queue, while it is already involved in an active connection with a certain tag.

After estimating the distances from the required number of anchors, the tag sends these measurements to the server. Upon receiving the distances, the server stores them locally, pushes the tag to the end of the queue and sends the “RECEIVED” acknowledgement back to the tag to confirm successful receipt. The server then repeats the same process with the next tag from the queue. This

[16]https://github.com/Razyapoo/Master-Thesis/blob/main/Implementation/Server/Server.cpp

cycle allows the server to handle each tag in a sequential order, resolving the issue where a single tag could monopolize the attention of the anchors.

Our own DS-TWR implementation for the centralized architecture

Remember that the code provided by the Makerfabs performs well, but only when there is a single operating tag in the system, as discussed in the Section 3.2.4. We hypothesized that integrating this with a centralized architecture might potentially eliminate this limitations.

To test this, we extended the code provided by Makerfabs to support message exchanges between the tags and the server. Subsequent testing revealed that this setup is still optimal only for a single operating tag. Attempts to add more tags to operate simultaneously led to communication issues, resulting in estimated distances being recorded as zero and sometimes even negative.

Eventually, we decided against overwriting the official library. Instead, we chose to adapt the code from the decentralized architecture.

As was mentioned at the beginning of this section, one significant disadvantage of the centralized architecture is that the frequency of the collected data is lower than in the decentralized architecture. To improve the communication speed, **we removed** all states that were used in the communication between anchors and tags, as well as between tags and the server. Instead the anchors and the tags always expect a certain message type from their counterparts as defined by the DS-TWR protocol.

DS-TWR protocol involves two round-trip exchanges to compute the distance between two devices, each consisting of two message transmissions. To indicate a certain step within a single round-trip exchange, both the anchor and the tag use specific message types. These types are identified by flags contented within the messages. Moreover, each device consistently verifies the address of the counterpart to prevent interruptions from other devices and to escape the problem with message overwriting, which is discussed in Section 3.2.4.

This changes allowed us to increase both the speed of the communication and, consequently, the frequency of the measurements collected from the tags.

One potential problem in the decentralized architecture, as well as in the official DW1000 library, is that each tag first performs a Discovery phase to identify the required number of anchors before initiating DS-TWR communication. This method poses a risk of deadlock when a person moves from one room to another, potentially loosing the signal from previously discovered anchors. In industrial environment this can happen often.

To address this issue, we eliminated the Discovery phase. Instead a tag directly initiates the Ranging phase by sending a "Poll" message to an anchor. This modification enhances the tag independence and mobility, allowing it to dynamically interact with any available anchor without being restricted to a specific set of anchors. This approach significantly reduces the time spent on the communication and the change of communication failures.

Once a tag receives the "Measure!" request from the server, it broadcasts a "Poll" message, including its address. Upon receiving the "Poll" message, the anchor records the tag's address together with the message's reception time. Then it waits for a specific delay. This setup raises a question about how to manage responses from multiple anchors. Simultaneous responses could overload the tag with incoming requests, causing it to freeze and preventing it from responding to the anchors.

In the decentralized system, this issue was managed by introducing random wait times before anchors responded to the tag, a process that occurred during the Discovery phase. Since we eliminated the Discovery phase in the centralized architecture, we needed a new solution.

We addressed the issue by establishing constant t_{reply} reply delays for each anchor. Experimentally, we found that using identical reply delays for all anchors caused signal collisions, preventing the tag from responding. This problem arose when we added a second and then a third anchor to the system. Eventually, we found optimal, distinct reply delays for each anchor to avoid collisions. These values were challenging to determine correctly, as many combinations led to collisions

When reply delay expires, the anchor responses to the tag with a "Poll ACK" message. Then the tag and the anchor communicate using the DS-TWR communication protocol. In the end, the tag calculates distances from the required number of anchors and sends them to the server, as discussed in the Section 3.3.4.

3.4 Evaluation of the collected UWB data

After the successful implementation of the DS-TWR protocol, we examined the precision of the UWB devices in distance estimation.

We performed experiments in four different environments: Dorm - ExpEnv1, Rot - ExpEnv2, S301 - ExpEnv3 and S8 - ExpEnv4.

3.4.1 Setting up the experiment environments

To evaluate the precision of distance estimations, we used the $\texttt{RS}_{\texttt{1m}}$ reference system, which was discussed in Section 3.2.1.

- **Dormitory - Experiment Environment 1**
  - **Description**: Simulates a narrow hallway of a manufacturing environment. The environment is shown in Figure 3.20.
  - **Challenges**: Includes reinforced concrete walls, causing signal interference, which significantly affects the accuracy of UWB measurements, as discussed in Section 3.4.3.
- **Rotunda - Experiment Environment 2**
  - **Description**: Simulates a small laboratory, equipped with large set of personal computers. The environment is shown in Figure 3.21.
  - **Challenges**: Involves the strong signal interference caused by large number of personal computers and other electronic devices.
- **S301 - Experiment Environment 3**
  - **Description**: Simulates a light version of a manufacturing environment. The environment is shown in Figure 3.22.

- **Challenges**: Includes cabinets that create obstacles, resulting in Non-line of Sight (NLoS) conditions. Additionally, it includes an electrical enclosure, creating signal interference, when the tag is located in close proximity to it.
- **Schema**: Schema is available in Figure A.1.

- **S8 - Experiment Environment 4**

  - **Description**: Simulates the long hallway, which is free of obstacles. The environment is shown in Figure 3.23.
  - **Challenges**: Includes cabinets that create obstacles, resulting in Non-line of Sight (NLoS) conditions. Includes an electrical enclosure, creating signal interference.
  - **Schema**: Schema is available in Figure A.2

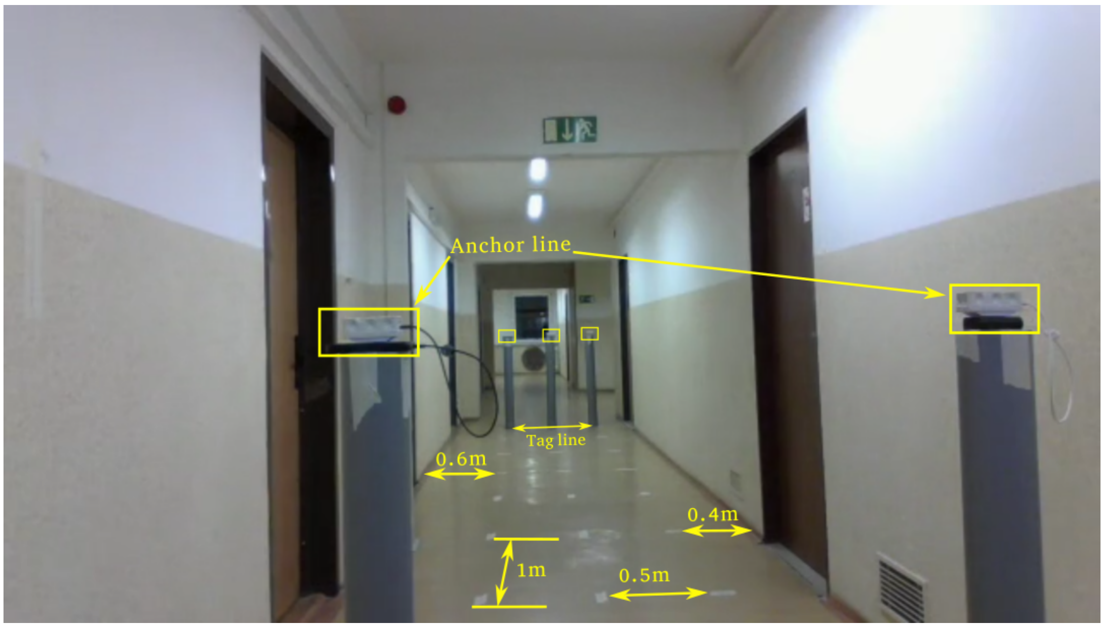


Figure 3.20 A photo of the Dorm - ExpEnv1. It simulates a narrow hallway of a manufacturing environment, including reinforced concrete walls that cause signal interference.

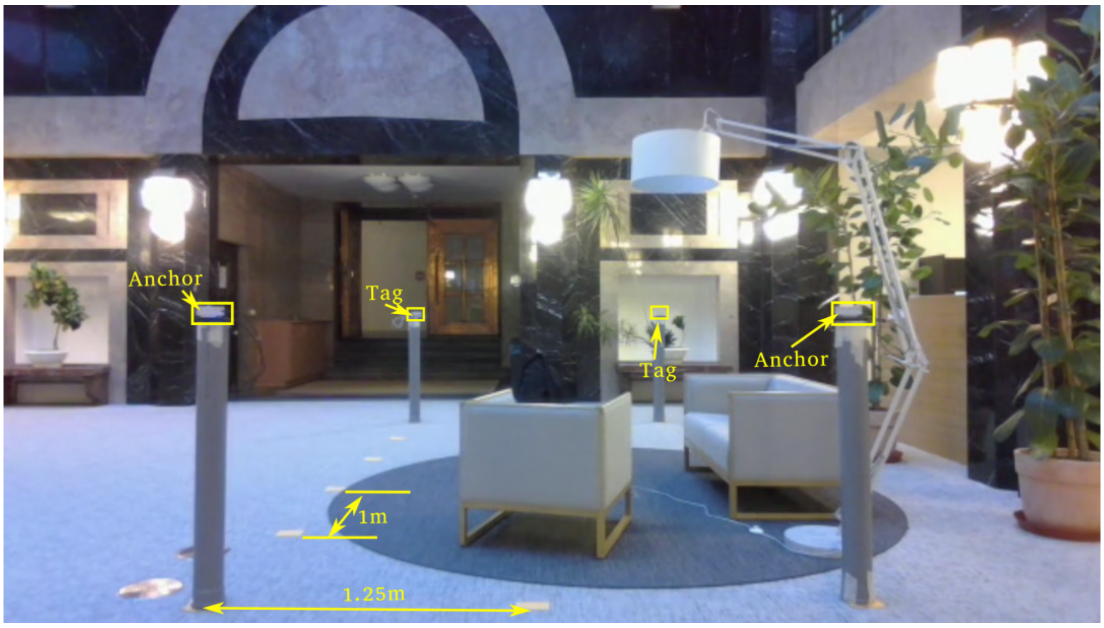


Figure 3.21 A photo of Rot - ExpEnv2. It simulates a small laboratory, equipped with large set of personal computers.

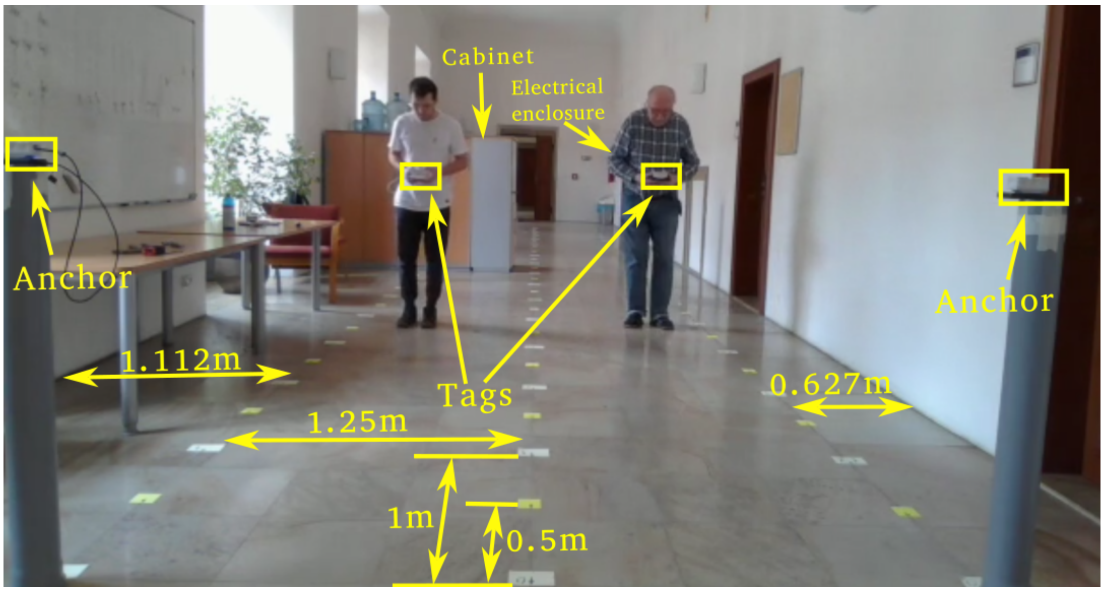


Figure 3.22 A photo of the S301 - ExpEnv3. It simulates a light version of a manufacturing environment. This environment is challenging due to the presence of cabinets and an electrical enclosure.

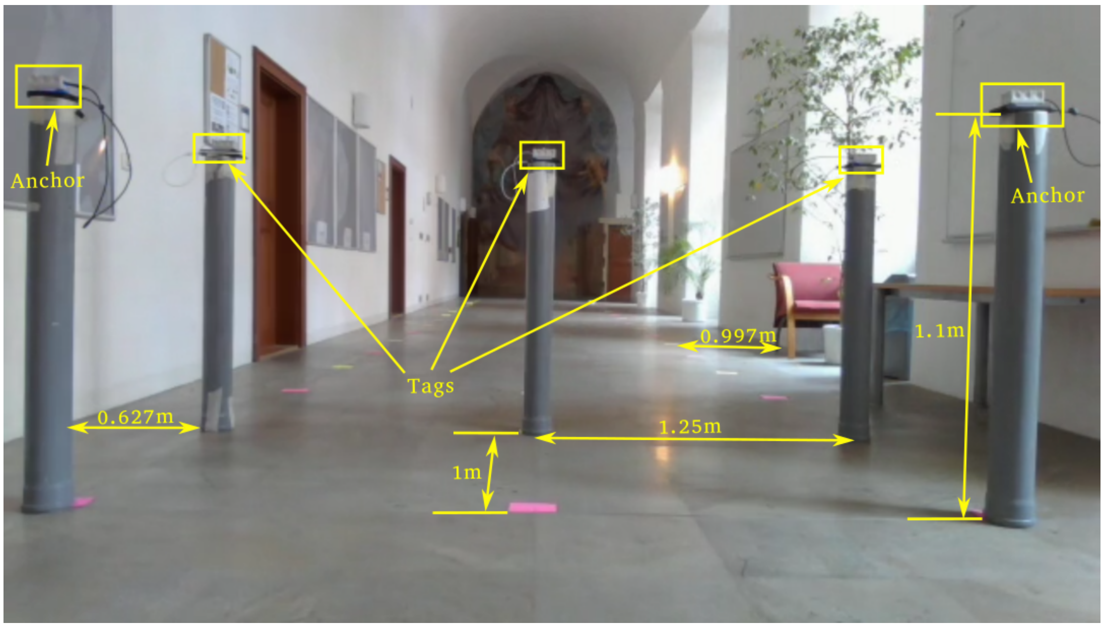


Figure 3.23 A photo of the S8 - ExpEnv4. It simulates the long hallway. This environment is challenging due to the presence of an electrical enclosure.

Plastic pipes as helping structure

Initially, during the experiments, we held the tags in our hands and walked through the experiment environment, stopping at each paper sticker for a certain period of time.

However, we discovered that **people** also **contribute to** the location **error** when holding a tag. Due to the natural variability of posture, they may not hold the tag exactly at the point where the sticker is, thus adding additional error to the estimated distance. Furthermore, people significantly contribute to the signal interference. When a person holds a tag close to his body, the signal is interrupted. We found that holding the tag slightly away from the body, without touching it, or above the head, does not affect the signal.

To address these problems, we used **plastic pipes**, as shown in Figure 3.23. We used them as stands for the tags, and placed them on each sticker for a certain period of time to measure the distance. The height of the pipes matches the height at which we held the tags. As the result, this allowed us to align the tags precisely with the locations of the stickers.

To precisely align the antennas of the anchors and tags, we used a pencil and thread and later a laser device.

3.4.2 Experiment scenarios for UWB evaluation

To examine the precision of the measured distances, we used several different scenarios aimed at measuring the distances and comparing them with the reference Ground Truth distances, given by $\texttt{RS}_{\texttt{1m}}$ reference system. The calibration of UWB devices was performed at an optimal distance of 5 meters. The reason is discussed in more detail in Section 3.4.4.

Furthermore, we conducted both static and dynamic experiments in all scenarios. In static experiments, the tags remained stationary, while in dynamic experiments, the tags were moved throughout the experiment environment.

In static experiments, we defined an anchor line for the anchors and a separate tag line for the tags, as shown in Figure 3.20. The tags and anchors stayed stationary along their corresponding lines. During the experiments, we shuffled the positions of the anchors and tags within their respective lines to ensure varied data collection. In the dynamic experiments, the anchors remained stationary, while the tags where moved independently of each other.

- **Scenario 1: Static tag calibration**
  - **Setup**: Configurations included various combinations of each tag and each anchor. For example, one tag with one anchor, one tag with two anchors, two tags with one anchor, two tags with two anchors, etc.
  - **Objective**: Calibrate and examine the accuracy of UWB measurements in static conditions.
- **Scenario 2: Dynamic tag calibration**
  - **Setup**: Used the same configurations as in Scenario 1, but added dynamic movements of the tags.

- **Objective**: Calibrate and evaluate the accuracy of UWB measurements under dynamic conditions.

- **Scenario 3: Obstacles**
  - **Setup**: Introduced obstacles while using various configurations of tags and anchors.
  - **Objective**: Assess the impact of obstacles and tag rotations on the accuracy of UWB measurements.

- **Scenario 4: Tag rotations**
  - **Setup**: Performing yaw rotations of tags.
  - **Objective**: Assess the impact of obstacles and tag rotations on the accuracy of UWB measurements.

- **Test Types**
  - **Direct Tests**: Distances are measured between the anchors and tags placed in the same rows. For example, the pair of Anchor 102 and Tag 3 shown in Figure 3.24.
  - **Diagonal Tests**: Distances are measured between the anchors and tags placed in different rows. For example, the pair of Anchor 101 and Tag 1.

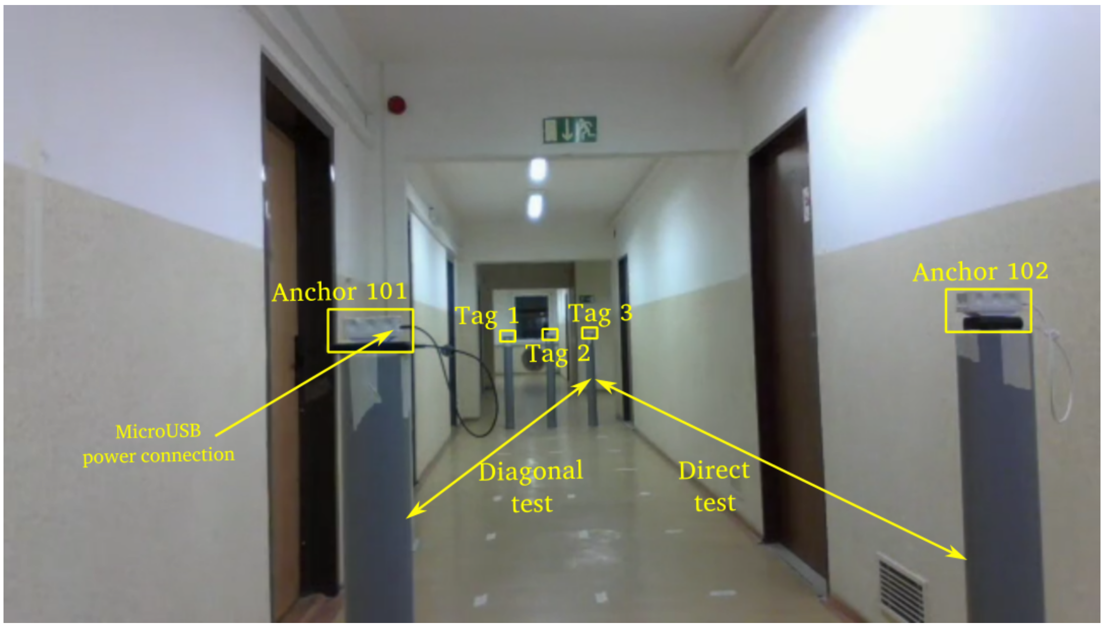


Figure 3.24 An example of direct and diagonal tests. This represents the E_{83} experiment.

3.4.3 Observations

During our experiments we have identified several anomalies. Some of these were expected, while others appeared unexpectedly.

During the experiments that follow the Scenario 1, we found that each tag and anchor requires its own calibrated antenna delay parameter, contrary to the suggestion of the official recommendations stated in DW1000 library. We sequentially placed each tag at the left, right, and middle sticker rows to measure distances and perform individual UWB device calibration. These experiments are available in the main repository of this thesis on GitHub[17].

Additionally, increasing the CPU frequency of UWB devices significantly improved communication speed. At 160 MHz, calculating distances from two anchors and sending them to the server took 250 milliseconds on average. Increasing the frequency to 240 MHz reduced this time to about 100-125 milliseconds.

Noise and signal interference

Following the Scenario 2, we discovered the discrepancy in the distance measurements. When the microUSB power connections of an anchor and a tag were directed towards each other, as in the case of Anchor 101 and Tag 3 shown in Figure 3.24, the signal was amplified, resulting in measured distances lower than the actual distances. Conversely, when the power connections were directed away from each other, as in the case of Anchor 102 and Tag 1, this led to signal attenuation, resulting in higher measured distances. Initial thought was that deviations in distances were caused by the devices themselves, because their lights were blinking. However, the problem was in the USB cables used for powering the UWB devices that acted as unintended antennas. They caused the signal dispersion effect: in the first case, the signal pulse remained strong due to the constructive interference, while in the latter case, signal dispersion reduced its strength.

During the experiments that follow the Scenario 3, we tested how the signal interacts with different obstacles. When signals pass through hands and chairs they do not make an interference. However, other objects, like walls, vegetation leafs and laptop can affect the signal, sometimes resulting in signal loss.

In addition, in the $\texttt{E}_{83}\texttt{(Calib\_Dorm\_S}_{3}\texttt{(T3\_A2\_TP}_{p}\texttt{\_M}_{d}\texttt{\_W}_{p}\texttt{))}$ experiment, we examined how people influence the signal propagation even when they do not hold any tag. When two tags and the anchor are placed in the same row, as shown on Figure 3.24, the signal is clearly received. However, when a person stands between tags, the tag behind the person loses the Line of Sight (LoS) with the anchor, causing a a connection break between the anchor and the tag. This situation is illustrated in Figure 3.25.

The orientation of the antennas matters. When the antennas of the anchor and the tag are directed away from each other, the measured distance increases and becomes unpredictable. This is the issue detailed in Section 3.1.2.

This observation was also confirmed in the $\texttt{E}_{6}\texttt{(Calib\_S301\_S}_{4}\texttt{(T1\_A2\_TP}_{h}\texttt{\_M}_{s}\texttt{\_W}_{p}\texttt{))}$ experiment, which examined tag rotations around the yaw axis. The tag showed increased measurements and lost connection when turning by more than 45 degrees with respect to the anchor.

Furthermore, we observed a signal interference near the electrical enclosure, as shown on Figure 3.26. In particular, the area of approximately 2 meters from the

[17]https://github.com/Razyapoo/Master-Thesis/tree/main/Recorded Experiments/Experiments (32 - 84) 2023.12.23 - 2023.12.30

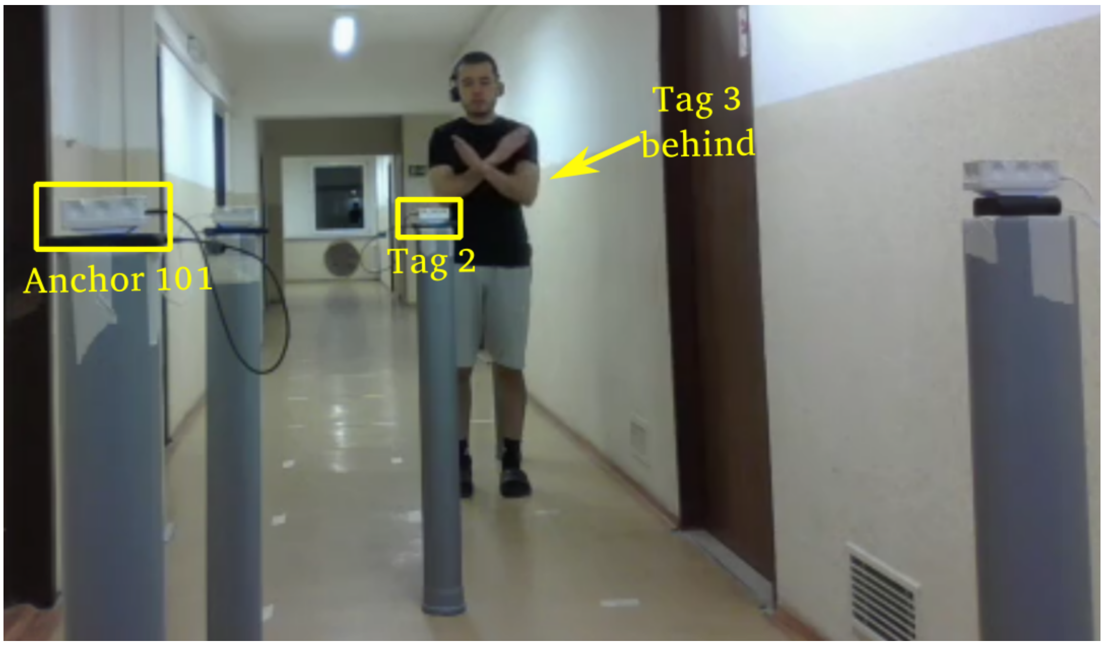


Figure 3.25 A person blocks the communication between the anchor and tag (behind person).

electrical enclosure was affected by the significant signal interference. In our initial versions of the DS-TWR implementation this interference resulted in signal loss. And in our latest implementations we observed its impact in affected distance measurements. This situation is analyzed in detail in the Chapter 7 and illustrated in Figure A.3.

Importance of the time delays during communication

Reply delay

As described in Section 3.3.1, during the DS-TWR communication, anchors and tags use reply delays to properly process the received signals. This delays prevent collisions that occur when multiple UWB devices start sending signals simultaneously, as discussed in Section 3.3.4. Although reply delays allows adjusting communication speed, very short delays can cause a loss of communication and require device resets.

Furthermore, the reply delay should be calibrated for each set of tags separately in order to avoid the collisions during the communication.

Hard resets

Even with correct reply delays, the anchors often stopped receiving signals from tags. We struggled with this problem for a long time without finding a solution. We added a watchdog to monitor if the anchor received or sent a signal within a certain period. If not, it performed a hard reset to resume the communication with the tag.

Experimentally we found that the optimal timeout value is 500 milliseconds,

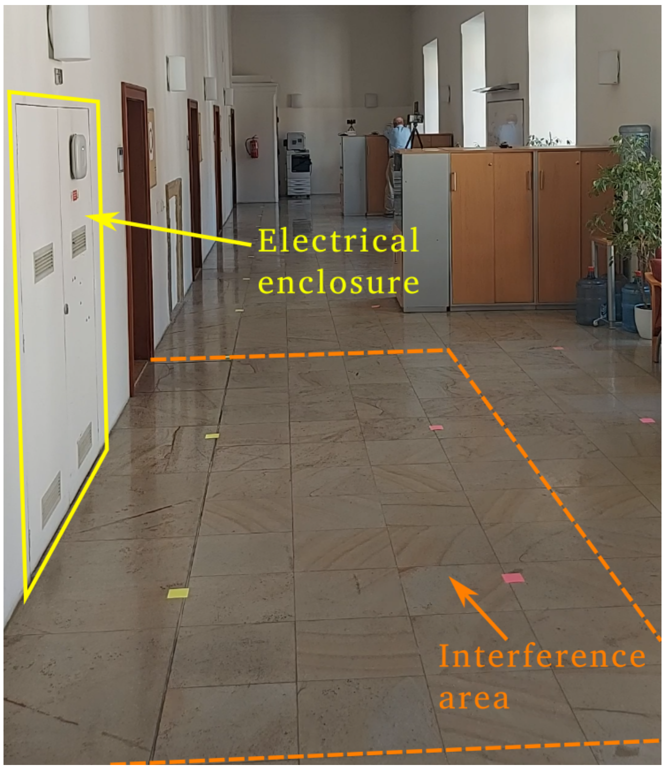


Figure 3.26 Interference area caused by electrical enclosure.

as a lower value could lead to indefinite cycling of the device, potentially causing it to **burn out**. This exact scenario occurred during our experiments.

3.4.4 Analysis of deviations in collected measurements

During the dynamic experiments, we noticed that the deviation in the measured UWB distances increased with the distance from the anchors, considering the calibration of the anchor and tag at a distance of 1 meter.

The Table 3.4 shows the deviation in measured distances compared to the Ground Truth distances given by $\texttt{RS}_{\texttt{1m}}$ reference system. The deviations are shown in Table 3.4.

Given this observation, we calibrated the anchors and tags at 5 meters, improving accuracy for both shorter and longer distances. Distances less than 5 meters showed a negative shift, while distances greater than 5 meters showed a positive shift. This allowed us to reduce the absolute error in distance estimation to approximately 0.12 meters, considering the same area size.

The **main goal** is to collect as **precise UWB coordinates** as possible, as this is mandatory for training the **precise Pixel-to-Real model**.

Therefore, we have trained polynomial regression model to correct the UWB distance estimations.

Resolving the deviations in data

Let's define (for more detail please refer to Appendix A):

- $D_{\text{raw}}(E_{\text{expid}})$ as the set of the **raw** distances collected during $\texttt{E}_{\texttt{expid}}$ experiment.

Measured distance	Reference distance
1	1.008
2	2.031
3	3.052
4	4.103
5	5.144
6	6.142
7	7.171
8	8.204
9	9.289

Table 3.4 An example of the discrepancy between the measured UWB distances and reference Ground Truth distances. It shows the average values of the estimated distances at each meter.

- $D_{\text{corr}}(E_{\text{expid}})$ as the set of the **corrected** distances collected during `Eexpid` experiment.

Since the measured data D_{raw} did not have a linear relationship, we implemented polynomial regression to correct measurements. We trained a function *CorrectUWBData* on known data and then used it to correct the measured distances, as follows:

$$D_{\text{corr}}(E_{\text{expid}}) = CorrectUWBData(D_{\text{raw}}(E_{\text{expid}}))$$

We stayed at each paper sticker for a certain period. This allowed us to detect standing periods using a standard rolling deviation, calculate the average distances for each period, and map them to the Ground Truth distances. This mapping helped to create the regression function, which we used to correct all measured distances, for both standing and moving periods.

The pseudocode of data correction function *CorrectUWBData* looks as follows:

Algorithm 4 Correcting UWB distances using polynomial regression

```
Input: Raw distance data $D_{raw}$, Ground Truth data $RS_{1m}$, window size $w$, deviation threshold $\delta$
Output: Corrected distance data $D_{corr}$
function CorrectUWBData($D_{raw}, RS_{1m}, w, \delta$)
    $rollingStd \leftarrow$ CalculateRollingStd($D_{raw}, w$)
    $stationaryPeriods \leftarrow$ DetectStationaryPeriods($D_{raw}, rollingStd, \delta$)
    $averages \leftarrow$ CalculateAverages($stationaryPeriods$)
    $regressionFunction \leftarrow$ TrainPolynomialRegression($averages, RS_{1m}$)
    $D_{corr} \leftarrow$ CorrectRawData($D_{raw}, regressionFunction$)
    return $D_{corr}$
end function
```

Instead of using rolling standard deviation, we also considered the use of ArUco markers[18]. Each marker is unique and therefore can be used to represent a specific

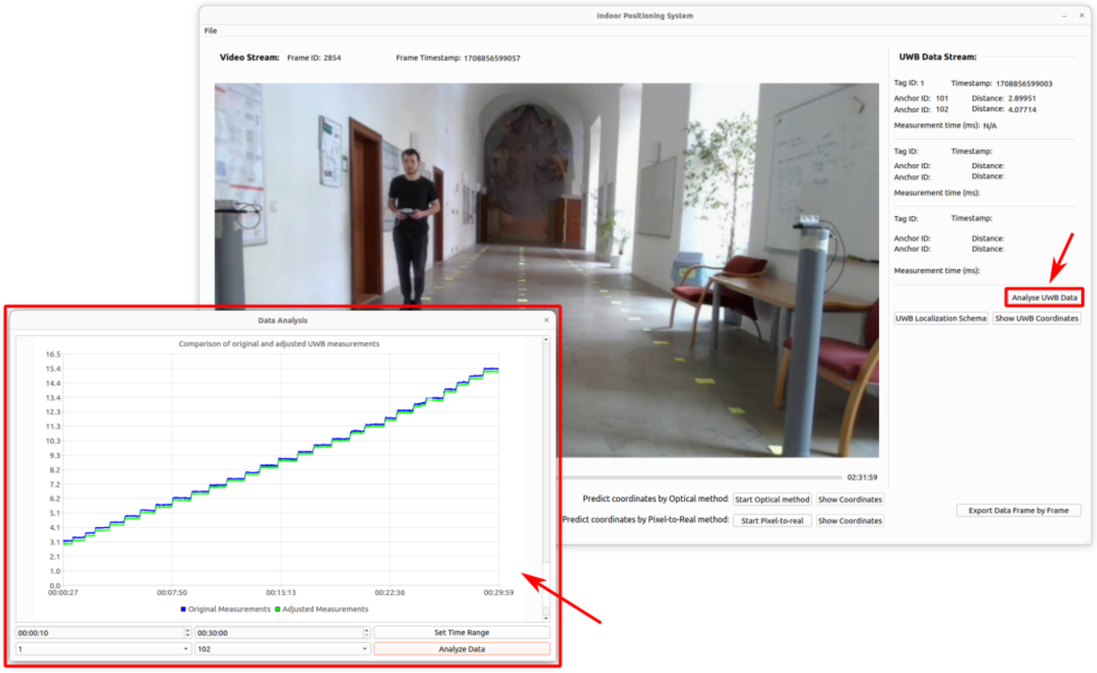
[18]https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

standing period. We held markers in front of the camera to later identify them during video processing. We have collected such data during $\mathtt{E}_7$-$\mathtt{E}_{14}$[19] experiments, but did not evolve this approach further.

The analysis of UWB data in GUI

The data analysis stage took us a considerable amount of time. Investigating the correct way of data analysis and correction involved extensive research. This has provided us with significant knowledge, which allows for faster deployment of our system for customers. We expect the deployment process to take approximately 1 or 2 days.

We integrated the capability to analyze and correct the collected UWB data into our *Indoor Positioning System* application. This analysis is performed in a dedicated window, as shown in Figure 3.27. The main purpose of this capability is to identify and correct anomalies in the collected data.


Indoor Positioning System

Video Stream: Frame ID: 2854 Frame Timestamp: 1708856599057

UWB Data Stream:

Tag ID: 1 Timestamp: 1708856599003
Anchor ID: 101 Distance: 2.89951
Anchor ID: 102 Distance: 4.07714
Measurement time (ms): N/A

Tag ID: Timestamp:
Anchor ID: Distance:
Anchor ID: Distance:
Measurement time (ms):

Tag ID: Timestamp:
Anchor ID: Distance:
Anchor ID: Distance:
Measurement time (ms):

Data Analysis

00:00:10 | 00:30:00 | Set Time Range
1 | 102 | Analyze Data

Figure 3.27 Data analysis window in *Indoor Positioning System* application.

The tool enables the users to select a specific time range for analysis. This feature is crucial for focusing on particular segments of data, allowing for more detailed data examination.

3.4.5 Calculating coordinates using UWB data

After estimating the UWB distances, we calculated people's coordinates using triangulation.

[19]https://github.com/Razyapoo/Master-Thesis/tree/main/Recorded Experiments/

Triangulation theory

We assume that people stay on the same floor when moving. Additionally, people hold tags at the same height as the anchors. This allows us to ignore the third dimension (height) in their coordinates.

Triangulation is a process of determining the location of a point (tag) by forming triangles to it from known points (anchors). It involves the two points of circle intersections (with centers at the anchors), defining two possible points where the tag can be located [26].

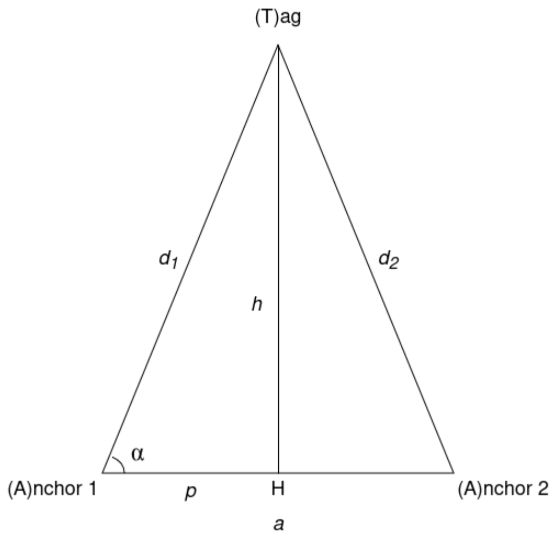


Figure 3.28 Triangulation setup with two anchors and one possible location of the tag. Another location can be found symmetrically.

Let's define the triangle formed by the anchors and tag, denoted as A_1A_2T (see Figure 3.28):

- A_1: *Anchor 1* with coordinates (x_1, y_1)
- A_2: *Anchor 2* with coordinates (x_2, y_2)
- T: *Tag* with coordinates $(x_{\text{UWB}}, y_{\text{UWB}})$
- A_1A_2: Anchor baseline, which is equal to a
- A_1T: Estimated UWB distance from *Tag* to *Anchor 1*, which is equal to d_1
- A_2T: Estimated UWB distance from *Tag* to *Anchor 2*, which is equal to d_2

Then the coordinates of the possible tag locations can be found as follows:

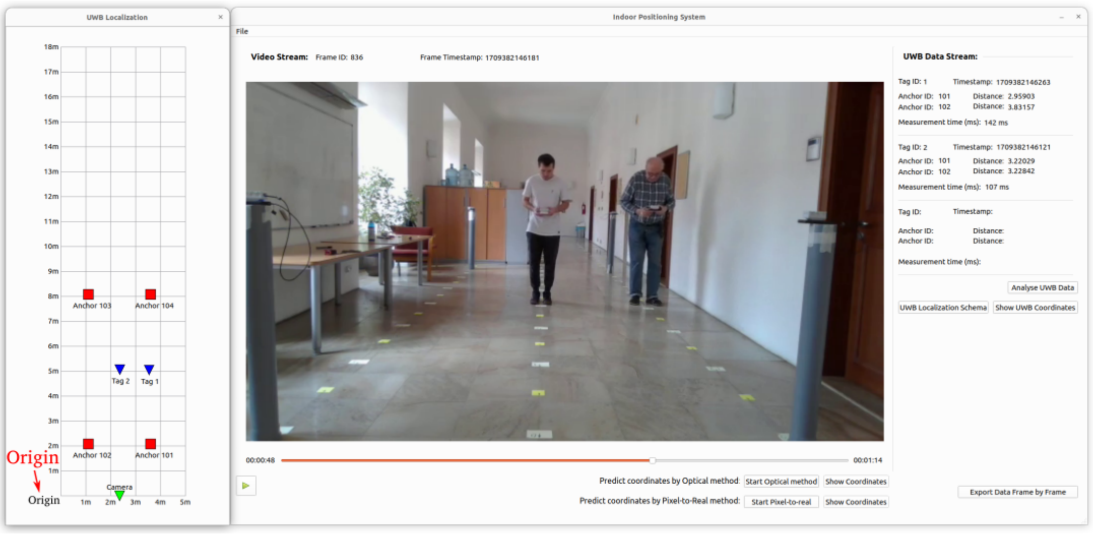
$$T_{1,2} = \left(x_1 + p \cdot \frac{x_2 - x_1}{a} \pm h \cdot \frac{y_2 - y_1}{a}, y_1 + p \cdot \frac{y_2 - y_1}{a} \mp h \cdot \frac{x_2 - x_1}{a} \right)$$

Assuming that the anchors always have known coordinates, we can easily select the relevant $(x_{\text{UWB}}, y_{\text{UWB}})$ coordinates for the *Tag*.

Visualization of the UWB people localization

Due to the specifics of the environments, such as the presence of cabinets or walls, it is often difficult to determine the exact locations of people from the video alone. To address this problem, we added a dedicated window to the *Indoor Positioning System* that displays a top-down scheme of the observed area. An example is shown on Figure 3.29.

Currently, we support a top-down view only for UWB localization.



Video Stream: Frame ID: 836 Frame Timestamp: 1709382146181

UWB Data Stream:

Tag ID: 1 Timestamp: 1709382146263
Anchor ID: 101 Distance: 2.95903
Anchor ID: 102 Distance: 3.83157
Measurement time (ms): 142 ms

Tag ID: 2 Timestamp: 1709382146121
Anchor ID: 101 Distance: 3.22029
Anchor ID: 102 Distance: 3.22842
Measurement time (ms): 107 ms

Tag ID: Timestamp:
Anchor ID: Distance:
Anchor ID: Distance:
Measurement time (ms):

Figure 3.29 UWB localization window in *Indoor Positioning System*. Red squares correspond to anchor locations, while blue triangles represent people.

3.4.6 Evaluation of UWB localization

We extended the coordinate system to cover the space between the anchor baseline and the camera filming the area. The origin of the real-world coordinate system is assumed to be at the left wall and camera line (see Figure 3.29). The camera is placed 2.08 meters behind the anchor baseline. The reason for shifting the coordinate system is to avoid sticking to the UWB coordinate system and extend it to cover the entire space of the environment, simulating real industrial conditions.

The evaluation of the estimated UWB coordinates is performed in Chapter 7 together with analysis of coordinates provided by Optical and Pixel-to-Real methods. This approach avoids repetition and simplifies the explanation of our analysis.

By utilizing the capabilities of our data analyzer, we ensured that the UWB coordinates are precisely estimated compared to the ground truth coordinates. This accuracy confirms that the UWB coordinates are reliable for further analysis and for developing the Pixel-to-Real predictive model, which is discussed in Chapter 5.

4 Synchronization between Video and UWB data

With correct distance measurements, we switched to the task of video and UWB synchronization. While the initial steps in synchronization are discussed in Section 3.2.2, this chapter is essential, as it represents the next step in developing the Pixel-to-Real model.

4.1 Timestamp-based synchronization

To improve synchronization, we switched to using webcams. This allowed us to assign absolute timestamps to each frame for more **accurate synchronization** with UWB data, which are also assigned with absolute timestamps. The timestamps correspond to Unix timestamps since Epoch, which makes them sequential and unique.

Video and UWB data are collected at different frequencies, which leads to mismatch in the timestamps. However, this can be resolved by matching each frame with UWB measurement that has the **closest timestamp**. This ensures that the matched data are collected as simultaneously as possible.

Let us define the following sets of data:

- $TS_{\text{video}} = \{t_{f(1)}, t_{f(2)}, \ldots, t_{f(n)}\}$ as the set of timestamps corresponding to video frames $f(i)$.
- $TS_{\text{UWB}} = \{t_{u(1)}, t_{u(2)}, \ldots, t_{u(m)}\}$ as the set of timestamps corresponding to uwb measurements.

where m and n are number of collected video frames and UWB measurements, respectively.

The search of the closest timestamp is performed as follows:

- For each $t_{f(i)} \in TS_{\text{video}}$, find $t_{u(k)} \in TS_{\text{UWB}}$, such that $\forall t_{u(j)} \in TS_{\text{UWB}} : |t_{f(i)} - t_{u(k)}| \leq |t_{f(i)} - t_{u(j)}|$

The UWB measurements are collected at lower frequency (4 Hz per tag) compared to the video (16 Hz). Additionally, anchors may periodically stop working and restart, as discussed in Section 3.4.3. This may further reduce the frequency of the collected UWB measurements, leading to potential gaps in the data. Moreover, message loss due to interference or other factors can also contribute to incomplete UWB data.

4.2 Adding video recording to the server

For more simultaneous synchronization, we introduced an additional thread to the server dedicated to video recordings. This thread is responsible for recording separate frames from the webcam and assigning them with the Unix timestamps since Epoch.

Additionally, we added a separate window, which allows us to track actual status of the recording. In particular, the green color indicates that the recording is being conducted without any problems, and the red color indicates that the recording has been interrupted. An example is shown in Figure 4.1.

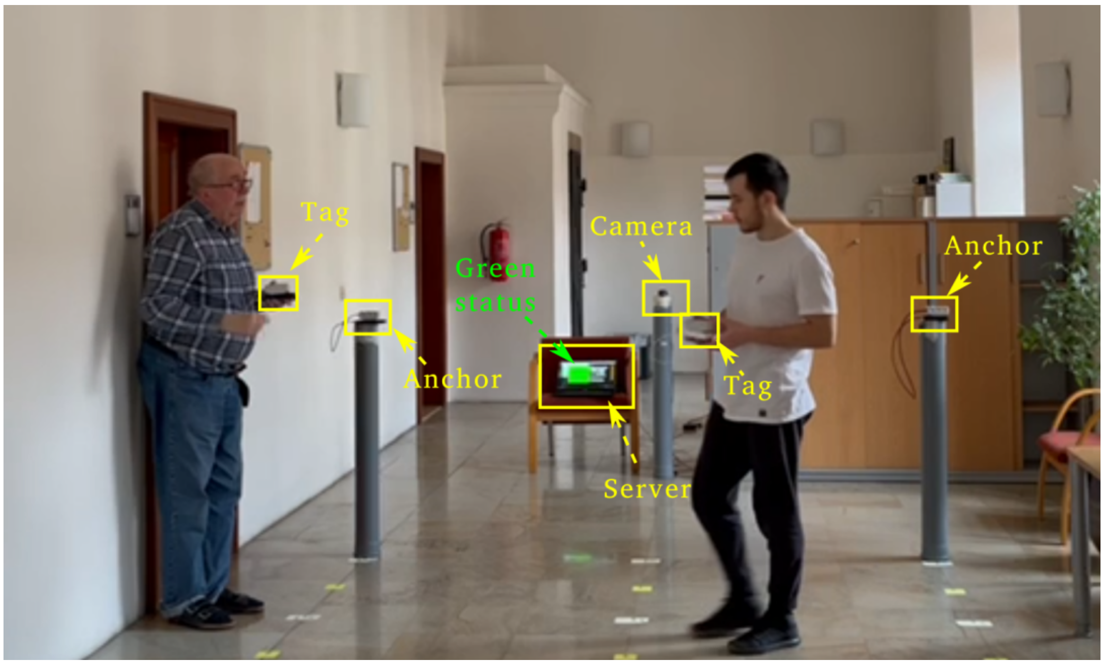


Figure 4.1 The green color indicates that the recording is being conducted without any problems.

5 Pixel-to-Real model

After collecting videos with annotated positions, we proceeded with training the machine learning model.

To uniquely determine people positions by the model, it is important to use a correct representation of input data. Additionally, the model should produce the same output for a given input every time.

The Convolutional Neural Network (CNN) can learn specific features in images and use them to accurately determine people locations. However, training a CNN typically requires high computational power.

On the other hand, a simpler models, such as linear regression or decision tree, allow us to train models faster and require less computational power, which is one of the most important criteria for many companies. These models can be highly effective and sufficient when the relationship between input and output data is straightforward. In our work we use decision tree as discussed later in Section 5.3.

5.1 Pixel coordinates as an input for the Pixel-to-Real model

We use the $(x_{\mathrm{p}}, y_{\mathrm{p}})$ pixel coordinates as an input for the Pixel-to-Real model to estimate the $(x_{\mathrm{P2R}}, y_{\mathrm{P2R}})$ real-world coordinates, as follows:

$$(x_{\mathrm{P2R}}, y_{\mathrm{P2R}}) = f(x_{\mathrm{p}}, y_{\mathrm{p}}) \tag{5.1}$$

where f represents the trained Pixel-to-Real function.

The $(x_{\mathrm{p}}, y_{\mathrm{p}})$ coordinates represent the center of the bottom edge of the bounding box detected by the YOLOv4 object detection framework. This choice is based on the fact that people have different heights, and the floor forms a common plane on which all people stand at the same level, as shown in Figure 5.1. This allows us to eliminate the third dimension (height) in real-world coordinates.

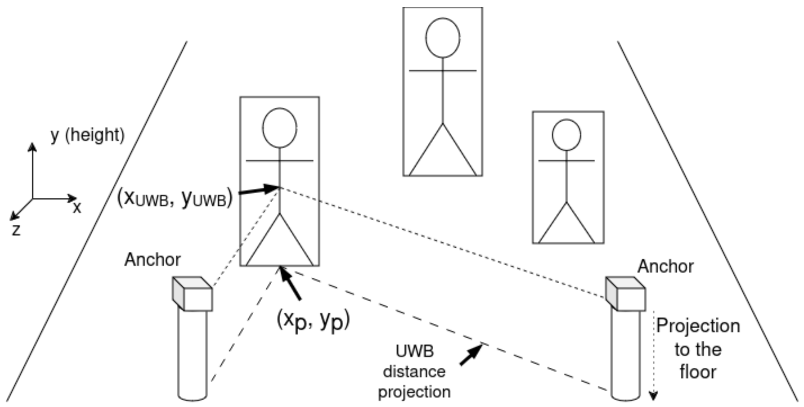


Figure 5.1 Projection of people coordinates onto the floor.

As discussed in Section 3.4.5, we hold tags at the same height as the anchors, which allows eliminate height when calculating $(x_{\mathrm{UWB}}, y_{\mathrm{UWB}})$ coordinates. This allows us to train the function f by matching the $(x_{\mathrm{p}}, y_{\mathrm{p}})$ and $(x_{\mathrm{UWB}}, y_{\mathrm{UWB}})$ coordinates, considering the UWB coordinates as a reference.

5.1.1 Alignment of detected people with UWB coordinates

As discussed in Section 4.1, for each video frame we find the UWB coordinates with the closest timestamp for each detected person. However, YOLOv4 produces detection boxes in the non-deterministic way, which leads to the problem of matching each person with the corresponding UWB coordinates. This problem can be solved by recognizing and tracking people to ensure the correct association between a tag and person. Recognition methods may include face recognition, clothing recognition, or behaviour analysis. People can also wear markers, such as Aruco markers, which makes the recognition process easier.

However, this process is challenging due to the difficulty of recognizing people in large areas, especially when cameras are positioned far from the people. Furthermore, tracking the people continuously is challenging task, especially when people mingle.

To address these challenges and to train the model accurately, we simplified the training data by ensuring that only one person is visible in the camera's field of view throughout the entire video. This approach guarantees that all measured UWB coordinates relate to a single person only, eliminating the need for complex tracking and identification algorithms. Furthermore, it is sufficient to collect the training data using only one person, because our model relies solely on the pixel coordinates of the bottom edge of the bounding box, and tag-person association is needed only for model evaluation.

Let us define the following sets of data:

- $P = \{(x_{p(1)}, y_{p(1)}), (x_{p(2)}, y_{p(2)}), \ldots, (x_{p(n)}, y_{p(n)})\}$ as the set of pixel coordinates in the video frames, corresponding to a single person. Each $(x_{p(i)}, y_{p(i)})$ coordinates correspond to the frame with the timestamp $t_{f(i)} \in TS_{\text{video}}$.
- $W = \{(x_{\text{UWB(1)}}, y_{\text{UWB(1)}}), (x_{\text{UWB(2)}}, y_{\text{UWB(2)}}), \ldots, (x_{\text{UWB(m)}}, y_{\text{UWB(m)}})\}$ as the set of real-world coordinates from UWB measurements, corresponding to a single person. Each $(x_{\text{UWB(i)}}, y_{\text{UWB(i)}})$ coordinates correspond to the frame with the timestamp $t_{u(i)} \in TS_{\text{UWB}}$.

The **timestamps** of the collected data serve as **unique identifiers** for that data. Thus, the timestamps can be used as means of aligning UWB coordinates with pixel coordinates. This alignment defines the following set:

$$A = \left\{ \left(\left(x_{p(i)}, y_{p(i)} \right), \left(x_{\text{UWB(k)}}, y_{\text{UWB(k)}} \right) \right) \middle| \; k = \operatorname*{argmin}_{j} |t_{f(i)} - t_{u(j)}| \right\} \tag{5.2}$$

5.2 Training the Pixel-to-Real model

To develop an accurate machine learning model for people localization, it is essential to split the dataset into distinct training, testing and validation sets. This ensures that the model is trained and evaluated on distinct datasets.

5.2.1 Splitting the dataset into training and testing sets

Several strategies existing for splitting the dataset, including by-frame splitting, segment-based splitting, and using separate videos for training and testing.

During by-frame splitting, every second frame of the video is used for training the model, and the remaining frames are used for testing. However, this can lead to model overfitting, because adjacent frames are highly similar. As a result, the model might perform bad on new, unseen data.

Segment-based splitting involves dividing the video into large, continuous segments and assigning them to the training, testing and validation sets. However, if these segments are similar in content, this also may result in overfitting.

To solve the problems associated with overfitting, we used **separate videos** for **training** and **testing** the model. To ensure the diversity in the data we have recorded videos using different:

- **Scenarios:**
  - **Scenario 5: Training and validation of the Pixel-to-Real model**

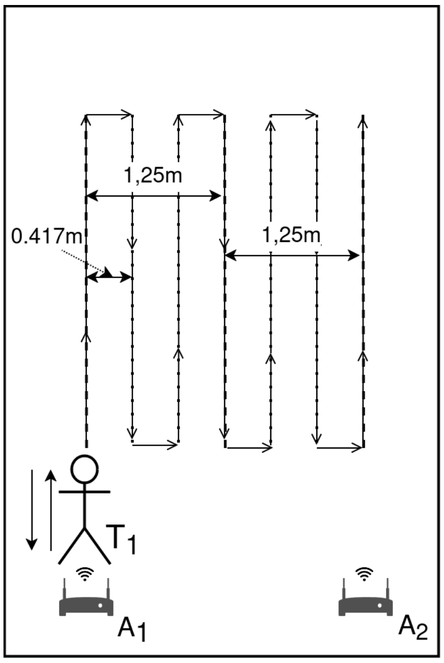


Figure 5.2 Scenario 5.

- **Objective**: To gather the extensive dataset for model training
- **Setup**: A person walks backwards along the entire length of the experiment environment, stopping every 0.5 meters for 30 seconds. Upon reaching the end of the line, the person moves to the line on the right and moves forward. The distance between lines is 0.417 meters. The process is repeated throughout the entire recording.
- Corresponding experiments are: $\mathtt{E}_{109}$-$\mathtt{E}_{117}$

  - **Scenarios 6-10:** Testing the Pixel-to-Real model.

- See Section 7.1 for more detail.
- Corresponding experiments are: $\mathtt{E}_{118}$-$\mathtt{E}_{132}$

- **Environments:** `S301`, `S8`.

– See Section 3.4.1 for more detail.

This approach minimizes the correlation between training, testing and validation data, providing a more accurate assessment of the model's performance.

5.3 Extreme Gradient Boosting framework for model training

While linear regression is a powerful regression model, it is not suitable for our goal, as the relationship between pixel coordinates and UWB coordinates is not linear. Furthermore, polynomial regression also resulted in overfitting, which led to inaccurate predictions on unseen data.

After further research, we chose Extreme Gradient Boosting (XGBoost). XGBoost is a machine learning algorithm that works based on gradient boosted decision tree [27]. This model proved to be more robust in handling the non-linear relationships between pixel and UWB coordinates and in handling the outliers and noise in the data. Handling the outliers is important, especially in the environments, where objects cause significant interference. An example of the interference caused by the electrical enclosure is discussed in Section 3.4.3.

The XGBoost represents the Pixel-to-Real function f (from Equation (5.1)). We train it on a dataset A (from Equation (5.2)).

Although, our *Indoor Positioning System* application does not provide direct capability to train the model, this feature will be added in the future work. Instead, it allows us to export the set A consisting of synchronized coordinates, providing flexibility in selecting a machine learning method. Furthermore, it supports the use of the trained model, enabling the application of the model to new unseen data.

The evaluation of the trained Pixel-to-Real model is provided in Chapter 7.

6 Optical people localization

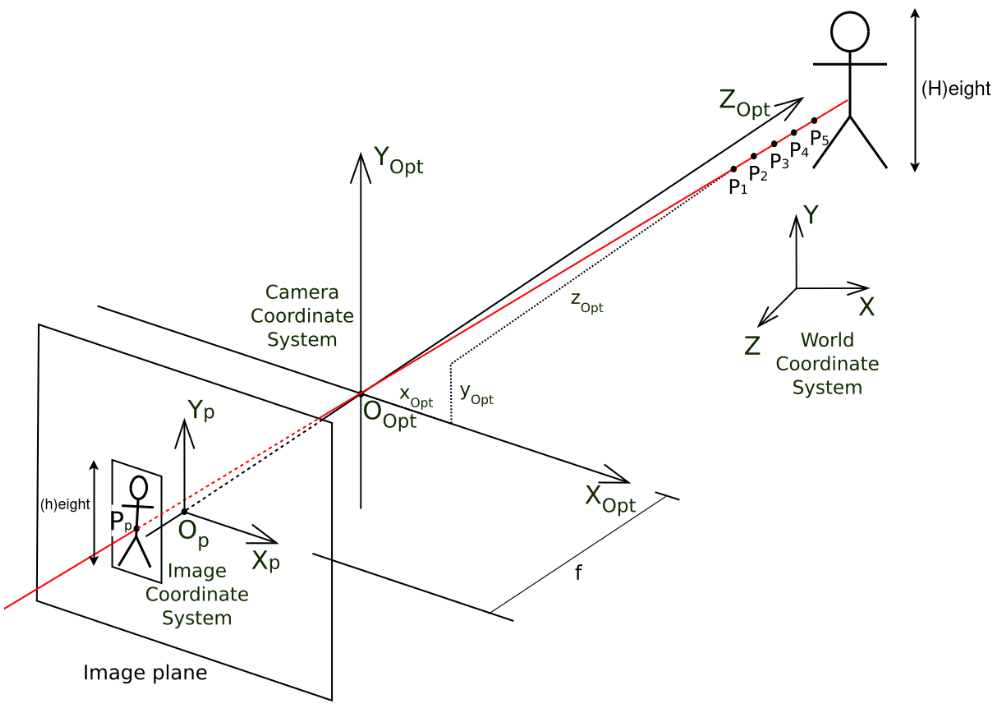


Figure 6.1 Projection of the 3D scene onto the 2D image plane. O_p represents the principal point of the camera, O_Opt represents the origin of the camera coordinate system, f is a camera's focal length.

In this chapter, we will describe the Optical method for people localization that we have implemented to further evaluate the Pixel-to-Real model. By implementing it, we also demonstrate the capability of our *Indoor Positioning System* application to evaluate other people localization methods.

A camera captures the real-world 3D scene as a 2D image using Planar Projection [28], resulting in a **loss of depth information**. For example , the points $P_1, P_2, \ldots, P_5$ project to the same point P_p on the image (camera sensor), as shown in Figure 6.1. Fortunately, it is possible to reconstruct the 3D coordinates from 2D image coordinates, if the distance (depth) from the camera to the observed person z_Opt and camera intrinsic parameters are known.

Similar to the UWB system, we expected to find an existing implementation of the 3D reconstruction from a 2D image. However, we have only found solutions that do not take into account the intrinsic camera parameters, resulting in very low accuracy in determining people locations. Therefore, we decided to implement our own Optical method for people localization, which relies on the height of the observed people and camera intrinsic parameters.

6.1 Height-based distance estimation

As discussed in Chapter 5, we detect people using YOLOv4 framework. Once people are detected, their distance from the camera is calculated based on their height in the image (height of the detection box in pixels) and the focal length f of the camera. This process requires precise calibration and understanding of the camera's parameters.

As discussed in Chapter 4, we use the webcam to capture the scene. We assume that **all its parameters are fixed**, including focal length, Field of View (FoV), rotation and orientation. Additionally, we use the already known heights of the observed people, and heuristic method to assign each person with his height. In the future, we plan to add a heuristic method to estimate people's heights without any prior knowledge.

The distance z_{Opt} can be calculated by the following formula:

$$z_{\mathrm{Opt}} = \frac{H \cdot f}{h} \tag{6.1}$$

where H is the actual height of the person, f is the focal length of the camera, and h is the height of the person in the image (see Figure 6.1).

This equation requires to know the focal length of the camera, which may vary for each camera, even for the same camera model from the same supplier. Therefore, it is mandatory to perform intrinsic calibration of the camera to get the correct focal length.

6.1.1 Intrinsic calibration

Modern cameras are not ideal, as they have the shift of the principal point. Fortunately, intrinsic calibration of the camera can help estimate this shift together with focal length.

The intrinsic parameters of a camera are typically represented by a matrix known as the Intrinsic Matrix K, which includes the focal lengths and the coordinates of the principal point [29]:

$$K = \begin{bmatrix} f_{\mathrm{x}} & 0 & c_{\mathrm{x}} \\ 0 & f_{\mathrm{y}} & c_{\mathrm{y}} \\ 0 & 0 & 1 \end{bmatrix}$$

where f_{x} and f_{y} are the focal lengths in the x and y directions, c_{x} and c_{y} are the coordinates of the principal point.

Non-linear intrinsic parameters such as **lens distortion** are also important to consider. Although, they are not included in the linear camera model described by the intrinsic parameter matrix, these parameters can be estimated using image rectification techniques [30].

Kenji Hata and Silvio Savarese provide an excellent explanation of the calibration and rectification processes in their CS231A Course[1].

With known depth information z_{Opt}, the real-world coordinates can be calculated as follows:

$$x_{\mathrm{Opt}} = (x_{\mathrm{p}} - c_{\mathrm{x}}) \cdot z_{\mathrm{Opt}} \cdot f_{\mathrm{x}} \tag{6.2}$$

$$y_{\mathrm{Opt}} = (y_{\mathrm{p}} - c_{\mathrm{y}}) \cdot z_{\mathrm{Opt}} \cdot f_{\mathrm{y}} \tag{6.3}$$

[1]https://web.stanford.edu/class/cs231a/course_notes.html

where $(x_{\mathrm{Opt}}, y_{\mathrm{Opt}}, z_{\mathrm{Opt}})$ are real-world coordinates of the observed object predicted by the Optical method (with respect to the camera coordinate system), $(x_{\mathrm{p}}, y_{\mathrm{p}})$ are image coordinates of the observed object, $(f_{\mathrm{x}}, f_{\mathrm{y}})$ are the focal lengths in the x and y directions, $(c_{\mathrm{x}}, c_{\mathrm{y}})$ are the coordinates of the principal point. Focal lengths and principal point are taken from the intrinsic parameter matrix.

6.2 Camera calibration

The calibration is performed using the OpenCV library and a 9×6 chessboard calibration pattern. This process involves capturing multiple images of the chessboard pattern from different angles and distances, which are then used to estimate the camera's intrinsic parameters. An example of the calibration process is shown in Figure 6.2.

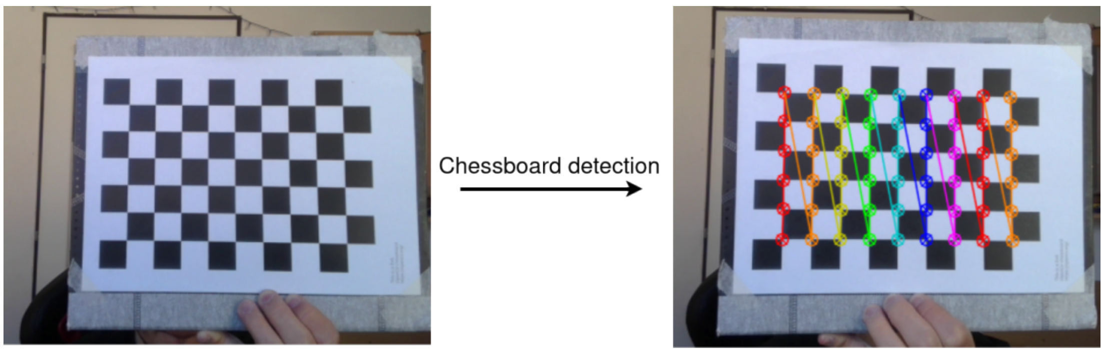


Figure 6.2 The process of the camera intrinsic calibration. Detection of the chessboard pattern.

6.3 Implementation of the Optical method

Intrinsic calibration only gives the coordinates relative to the camera's coordinate system. To get the coordinates in the world coordinate system, it is necessary to apply a rotation and translation transformations.

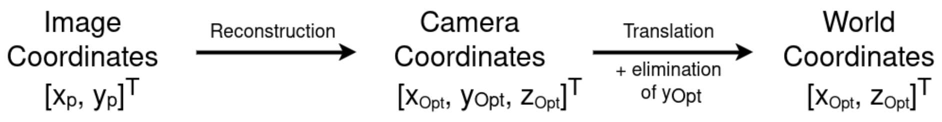


Image Coordinates $[x_p, y_p]^T$
Reconstruction
Camera Coordinates $[x_{Opt}, y_{Opt}, z_{Opt}]^T$
Translation
+ elimination of y_{Opt}
World Coordinates $[x_{Opt}, z_{Opt}]^T$

Figure 6.3 Pipeline of the coordinates' reconstruction.

When converting camera coordinates to real-world coordinates, we consider only translation, which allows to extend the coordinate system to cover the entire space of the environment, simulating real industrial conditions, as discussed in Section 3.4.6.

Additionally, to align with the coordinates provided by the UWB and Pixel-to-Real methods, we eliminate the height dimension in the Optical method. Furthermore, we assume that people always stand on the same floor (at the same height). Therefore, we select **only** $(\boldsymbol{x_{\mathrm{Opt}}}, \boldsymbol{z_{\mathrm{Opt}}})$ coordinates.

The evaluation of the Optical method is provided in Chapter 7.

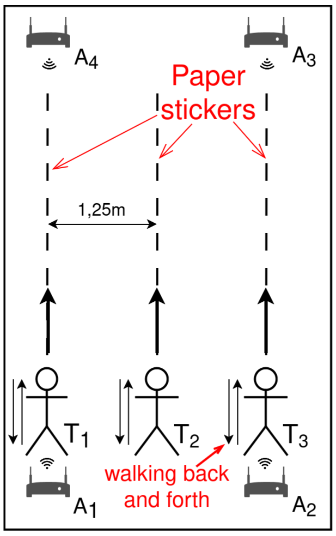
7 Evaluation and comparison of people localization methods

In this section, we will evaluate and compare the UWB, Pixel-to-Real and Optical methods. In the end, we will highlight the strengths and limitations of each method.

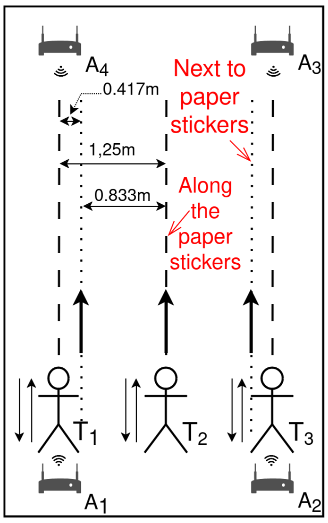
7.1 Experiments scenarios

During experiments, we followed distinct scenarios to evaluate the performance of our localization methods. It is important to note that in these scenarios we used four anchors placed in pairs on opposite sides from each other, as shown in Figure 7.1. All scenarios consider clear Line of Sight (LoS), without obstacles.

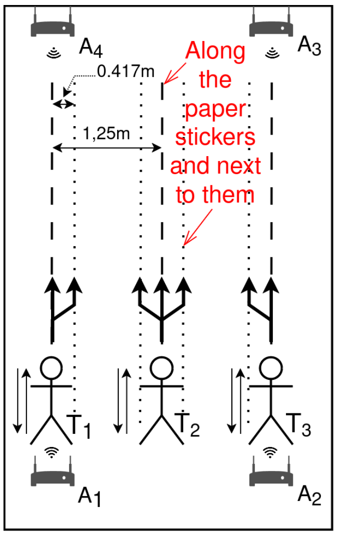
- **Scenario 6: Walking solely along the stickers.**
  - **Objective**: Test the accuracy of distance measurements and estimate people's coordinates. This experiment is similar to the Scenario 2 described in Section 3.4.1, but involve three people instead of using plastic pipes.
  - **Setup**: Three people synchronously walk along the paper sticker rows, stopping at each paper sticker for one second. They walk in both directions, towards Anchor 1 and Anchor 2 pair, and Anchor 3 and Anchor 4 pair. Each person walks solely along a certain sticker row.
  - **Complexity**: Evaluation of measured distances and estimated coordinates is straightforward, as the distance to each paper sticker is known.
- **Scenario 7: Walking along the stickers and next to them. Row switching is forbidden.**
  - **Objective**: Observe the possible deviation in measurements and try to estimate people's coordinates.
  - **Setup**: The setup is similar to the Scenario 6. In addition, the first person (from the left) walks next to his sticker row on the inner side, toward the middle person. The second person walks solely on his sticker row. The third person also walks next to his sticker row on the inner side, toward the middle person. It is only allowed to walk right on the predefined row and forbidden to change rows.
  - **Complexity**: Evaluation of estimated people's locations is more challenging, as tags are located closer to each other, potentially causing signal interference. Additionally, tags and anchors are not aligned on a straight line.
- **Scenario 8: Walking along the stickers and next to them. Row switching is allowed.**



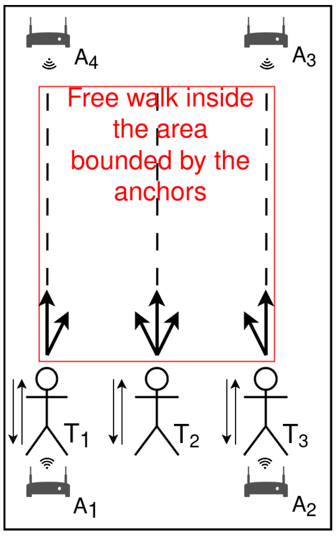
(a) Scenario 6



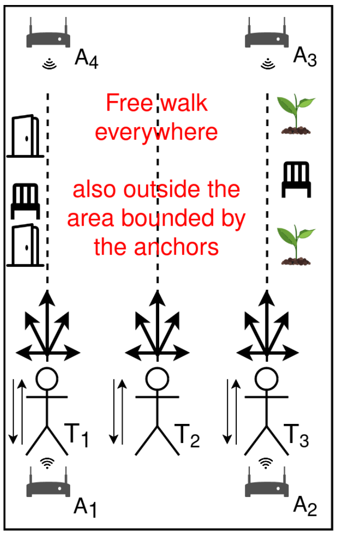
(b) Scenario 7



(c) Scenario 8



(d) Scenario 9.



(e) Scenario 10.

Figure 7.1 Experiment scenarios aimed at testing the developed methods.

  - **Objective**: Test the accuracy of distance measurements and observe the possible deviation in measurements, particularly when tags are rotating and causing signal interference. Then try to estimate people's coordinates.
  - **Setup**: The setup is similar to Scenario 7. In contrast, people can walk asynchronously. Additionally, the first person (from the left) walks along the predefined sticker row and next to to it from the inner side, toward the middle person. The second person walks along the predefined row and is allowed to walk next to the row from both sides. The third person walks the same as the first person, but from another side.
  - **Complexity**: People may walk asynchronously in any direction, potentially creating Non-line of Sight (NLoS) for other tags. Additionally, when switching between rows, tag rotations can cause signal interference with other tags and affect measurement accuracy, emulating the Scenario 4 described in Section 3.4.1, but with three people.

- **Scenario 9: Free walk. Only inside the area bounded by four anchors.**
  - **Objective**: Simulate more natural movement patterns and test the behavior.
  - **Setup**: Three people walk freely within the area bounded by the four anchors in different directions, without following a predefined path.
  - **Complexity**: This scenario presents a more difficult task as the nature of the movement is unpredictable. This results in a more challenging evaluation of the accuracy of distance measurements and estimated people's coordinates.

- **Scenario 10: Free walk everywhere. Inside and outside the area bounded by four anchors.**
  - **Objective**: Simulate even more natural movements under more natural conditions.
  - **Setup**: The setup is similar to Scenario 9, but people can walk outside the area bounded by the four anchors. They are allowed to sit on chairs, walk to the plants and enter the doors.
  - **Complexity**: This scenario presents the most challenging experiment. The nature of movements is unpredictable, and the presence of additional objects like chairs, plants, and doors further contributes to signal interference.

We perform our evaluation solely based on the Scenario 5 and Scenario 6.

7.2 Experiments aimed at evaluating the developed methods

The evaluation is performed on the data collected during the following experiments. Raw data are available on GitHub[1]:

[1]https://github.com/Razyapoo/Master-Thesis/tree/main/Recorded Experiments

- $\mathtt{E_{109}(DA\_S8\_S_5(T1\_A2\_TP_h\_M_d\_W_{pn}))}$

  This experiment simulates the system's calibration process at the deployment stage. The data collected during this experiment is used to train the Pixel-to-Real model, due to the large area of the S8 - ExpEnv4 environment. In this scenario a person walks slowly, stopping at each paper sticker (Scenario 5). This allows for extensive analysis of the collected data.

- $\mathtt{E_{113}(DA\_S301\_S_5(T1\_A2\_TP_h\_M_d\_W_{pn}))}$

  This experiment simulates a simple scenario with a single person and allows for the evaluation of the model trained on the data collected in $\mathtt{E_{109}}$. This experiment is conducted in an environment S301 - ExpEnv3. This environment differs from the one on which the model is trained, making the evaluation especially interesting as it tests the model's performance in a new environment. Although the data collected during this experiment can also be used to calibrate our system and train the Pixel-to-Real model, we let this task for the future work. This experiment can be considered as the one that is used during the system's calibration process at the deployment stage.

- $\mathtt{E_{118}(DA\_S8\_S_6(T3\_A4\_TP_h\_M_d\_W_p))}$

  This experiment simulates more realistic conditions and aims to demonstrate the performance of our system in the operational stage. It involves the walking of three people at almost normal walking speed and is recorded in the same environment as $\mathtt{E_{109}}$. For evaluation, we use the model trained on data from $\mathtt{E_{109}}$.

- $\mathtt{E_{124}(DA\_S301\_S_6(T2\_A4\_TP_h\_M_d\_W_p))}$

  Similar to $\mathtt{E_{118}}$, this experiment simulates realistic conditions to evaluate our system's performance in the operational stage, using the model trained on data from $\mathtt{E_{109}}$. It involves the walking of 2 people at almost normal walking speed. This experiment is conducted in the S301 - ExpEnv3 environment, which is different from the one on which the model is trained.

This notation represents the 109th, 113th, 118th and 124th experiments. The type of experiments is data acquisition `DA`, conducted in the environments `S8` and `S301`. The $\mathtt{E_{109}}$ and $\mathtt{E_{113}}$ experiments follow the scenario $\mathtt{S_5}$ (see Section 5.2), while the $\mathtt{E_{118}}$ and $\mathtt{E_{124}}$ experiments follow the scenario $\mathtt{S_6}$. The $\mathtt{E_{109}}$ and $\mathtt{E_{113}}$ experiments involve the use of 1 tag and 2 anchors. The $\mathtt{E_{118}}$ experiment involves the use of 3 tags and 4 anchors. The $\mathtt{E_{124}}$ experiment involves the use of 2 tags and 4 anchors. Tags are placed on hands ($\mathtt{TP_h}$) in each experiment. The experiments involve dynamic movement ($\mathtt{M_d}$). The $\mathtt{E_{109}}$ and $\mathtt{E_{113}}$ experiments involve walking pattern both on predefined lines and next to them ($\mathtt{W_{pn}}$), while the $\mathtt{E_{118}}$ and $\mathtt{E_{124}}$ experiments involve walking pattern on predefined lines ($\mathtt{W_p}$) only. During these experiments, we have collected $\mathtt{D_{raw}(E_{109})}$, $\mathtt{D_{raw}(E_{113})}$, $\mathtt{D_{raw}(E_{118})}$, $\mathtt{D_{raw}(E_{124})}$ sets of distances. Please refer to Appendix A for more detail about this notation.

The UWB method provides a set of coordinates that are calculated specifically for each experiment from $\mathtt{D_{corr}(E_{109})}$, $\mathtt{D_{corr}(E_{113})}$, $\mathtt{D_{corr}(E_{118})}$, $\mathtt{D_{corr}(E_{124})}$ sets of distances. A $\mathtt{D_{corr}(E_{expid})}$ set of distances is derived from $\mathtt{D_{raw}(E_{expid})}$ by correction of raw distances using polynomial regression, as described in Section 3.4.4. The set of coordinates provided by the UWB method is further denoted as $\mathtt{C_{UWB}(E_{109})}$ for experiment $\mathtt{E_{109}}$, $\mathtt{C_{UWB}(E_{113})}$ for experiment $\mathtt{E_{113}}$, $\mathtt{C_{UWB}(E_{118})}$ for experiment $\mathtt{E_{118}}$ and $\mathtt{C_{UWB}(E_{124})}$ for experiment $\mathtt{E_{124}}$.

The Pixel-to-Real method provide a set of coordinates that are predicted by the Pixel-to-Real model, as described in Chapter 5. It is important to note that the Pixel-to-Real model is trained on the $\mathtt{C_{UWB}(E_{109})}$ coordinates calculated based on $\mathtt{D_{corr}(E_{109})}$ set of the **corrected** distances. However, similar to the UWB and Optical methods, the Pixel-to-Real method is tested in both S8 - ExpEnv4 and S301 - ExpEnv3 environments. This allows to evaluate the model's performance and demonstrate its effectiveness. The set of coordinates provided by the Pixel-to-Real method is further denoted as $\mathtt{C_{P2R}(E_{109})}$ for experiment $\mathtt{E_{109}}$, $\mathtt{C_{P2R}(E_{113})}$ for experiment $\mathtt{E_{113}}$, $\mathtt{C_{P2R}(E_{118})}$ for experiment $\mathtt{E_{118}}$ and $\mathtt{C_{P2R}(E_{124})}$ for experiment $\mathtt{E_{124}}$.

The Optical method provide a set of coordinates that are calculated using the Equation (6.3). This set of coordinates is further denoted as $\mathtt{C_{Opt}(E_{109})}$ for experiment $\mathtt{E_{109}}$, $\mathtt{C_{Opt}(E_{113})}$ for experiment $\mathtt{E_{113}}$, $\mathtt{C_{Opt}(E_{118})}$ for experiment $\mathtt{E_{118}}$ and $\mathtt{C_{Opt}(E_{124})}$ for experiment $\mathtt{E_{124}}$.

People detection is performed on the undistorted images for all methods. Undistortion is performed using the intrinsic camera parameters.

The analysis of $\mathtt{E_{118}}$ and $\mathtt{E_{124}}$ experiments is very difficult and time consuming, because they assume more realistic conditions. In particular, these experiments involve more people as participants. Furthermore, during these experiments, the participants walk faster, which makes post-processing of the collected data more difficult and complex.

Assuming that UWB method is used only during the calibration process of our localization system and only at the deployment stage, we can conclude that during deployment, it is much more effective to perform experiments like $\mathtt{E_{109}}$ and $\mathtt{E_{113}}$, which allow for more precise system calibration. On the other hand, such experiments should include more participants to enhance the accuracy of the calibration.

During this analysis, we have created around 120 different plots and around 50 tables with statistics, which visualize the calculated statistics and metrics. Not all of them are shown in this work, but only the most interesting and informative ones. However, all these plots and statistics are available on GitHub[2].

7.3 Selecting coordinates for the evaluation

The evaluation of the estimated coordinates involves their comparison with the reference Ground Truth coordinates. However, this is a complex task, as it requires the knowledge of the Ground Truth coordinates for every position within the entire environment.

To address this, we utilize the capabilities of our *Indoor Positioning System* application. As was discussed Section 3.4.4, it allows to detect time periods where a person is standing.

[2]https://github.com/Razyapoo/Master-Thesis/tree/main/PixelToReal, Optical and UWB evaluation/Relusts of evaluation (Plots, Statistics)

During the collection of data, necessary for Pixel-to-Real model, we force people to stand on the sticker papers for which the coordinates are known, providing us with the reference Ground Truth coordinates.

For each detected standing period, we calculate an average of the coordinates within that period, and use it as a single representative for comparison with Ground Truth coordinates. This helps to smooth out the estimated coordinates and mitigates the risk of selecting an outlier with a high error as the representative value.

For evaluation, we use a $\mathtt{RS}_{\mathtt{0.5m}}$ reference system with adjusted spacing between the stickers within each row to a 0.5 meters to evaluate the developed methods more precisely.

7.4 Comparative evaluation of coordinates

In this section we will perform the comparative evaluation of the coordinates provided by UWB, Pixel-to-Real and Optical methods.

7.4.1 Evaluation based on Ground Truth coordinates

In this section, we will perform the evaluation of the coordinates obtained using UWB, Pixel-to-Real and Optical methods by comparing them with the Ground Truth reference system $\mathtt{RS}_{\mathtt{0.5m}}$ (paper stickers). This evaluation aims to determine the absolute accuracy of each localization method.

We use the following notation to express the error in coordinates estimation for each method:

$$\mathtt{Error}(\mathtt{C}_{\{\mathtt{method}\}},\ \mathtt{RS}_{\mathtt{0.5m}}) = \begin{cases} |\mathtt{x}_{\{\mathtt{method}\}} - \mathtt{x}_{\mathtt{ref}}|, & \text{for } \mathtt{x} \text{ coordinate} \\ |\mathtt{y}_{\{\mathtt{method}\}} - \mathtt{y}_{\mathtt{ref}}|, & \text{for } \mathtt{y} \text{ coordinate} \end{cases} \tag{7.1}$$

where `method` can acquire one of the following values: `UWB, P2R, Opt`.

A pair $(\mathtt{x}_{\mathtt{UWB}},\ \mathtt{y}_{\mathtt{UWB}})$ represents the coordinates obtained using the UWB method; a pair $(\mathtt{x}_{\mathtt{P2R}},\ \mathtt{y}_{\mathtt{P2R}})$ represents the coordinates obtained using the Pixel-to-Real method; a pair $(\mathtt{x}_{\mathtt{Opt}},\ \mathtt{y}_{\mathtt{Opt}})$ represents the coordinates obtained using the Optical method; and a pair $(\mathtt{x}_{\mathtt{ref}},\ \mathtt{y}_{\mathtt{ref}})$ represents the reference Ground Truth coordinates within the $\mathtt{RS}_{\mathtt{0.5m}}$ reference system.

The calculated error metrics are provided in:

- Table 7.1 for the $\mathtt{E}_{109}(\mathtt{DA\_S8\_S}_5(\mathtt{T1\_A2\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{pn}))$ experiment, considering the entire area of the environment, which is 17.08 meters long.
- Table 7.2 for the $\mathtt{E}_{113}(\mathtt{DA\_S301\_S}_5(\mathtt{T1\_A2\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{pn}))$ experiment. In this experiment we consider only the entire range of the experiment, which is 8.58 meters long.
- Table 7.3 for the $\mathtt{E}_{118}(\mathtt{DA\_S8\_S}_6(\mathtt{T3\_A4\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{p}))$ experiment, considering the entire area of the environment, which is 17.08 meters long.

- Table 7.4 for the $\texttt{E}_{124}\texttt{(DA\_S301\_S}_6\texttt{(T2\_A4\_TP}_\texttt{h}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{p}\texttt{))}$ experiment. In this experiment we consider only the entire range of the experiment, which is 8.08 meters long.
- Table 7.5 for the $\texttt{E}_{109}\texttt{(DA\_S8\_S}_5\texttt{(T1\_A2\_TP}_\texttt{h}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{pn}\texttt{))}$ experiment, considering the reduced area of the environment, which is 10.08 meters long.
- Table 7.6 for the $\texttt{E}_{118}\texttt{(DA\_S8\_S}_6\texttt{(T3\_A4\_TP}_\texttt{h}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{p}\texttt{))}$ experiment, considering the reduced area of the environment, which is 10.08 meters long.

Method	Coordinate	*MAE*	*MSE*
Error($C_{UWB}(E_{109}$, Tag 1), $RS_{0.5m}$(Tag 1))	$\texttt{x}_\texttt{UWB}$	0.0997	0.0264
	$\texttt{y}_\texttt{UWB}$	0.0248	0.0013
Error($C_{P2R}(E_{109}$, Tag 1), $RS_{0.5m}$(Tag 1))	$\texttt{x}_\texttt{P2R}$	0.099	0.023
	$\texttt{y}_\texttt{P2R}$	0.0855	0.0169
Error($C_{Opt}(E_{109}$, Tag 1), $RS_{0.5m}$(Tag 1))	$\texttt{x}_\texttt{Opt}$	0.3445	0.1441
	$\texttt{y}_\texttt{Opt}$	0.2059	0.0839

Table 7.1 Comparative evaluation metrics showing the discrepancy between estimated and reference Ground Truth coordinates (pseudo-code: `error = |c{method} - cref|`, where `c = {x, y}`). These metrics are calculated based on data collected in $\texttt{E}_{109}$ experiment, **considering the entire area of the experiment, which is 17.08 meters long**.

Method	Coordinate	*MAE*	*MSE*
Error($C_{UWB}(E_{113}$, Tag 1), $RS_{0.5m}$(Tag 1))	$\texttt{x}_\texttt{UWB}$	0.0187	0.0006
	$\texttt{y}_\texttt{UWB}$	0.01	0.0002
Error($C_{P2R}(E_{113}$, Tag 1), $RS_{0.5m}$(Tag 1))	$\texttt{x}_\texttt{P2R}$	0.075	0.0123
	$\texttt{y}_\texttt{P2R}$	0.1656	0.0553
Error($C_{Opt}(E_{113}$, Tag 1), $RS_{0.5m}$(Tag 1))	$\texttt{x}_\texttt{Opt}$	0.1993	0.0452
	$\texttt{y}_\texttt{Opt}$	0.183	0.0477

Table 7.2 Comparative evaluation metrics showing the discrepancy between estimated and reference Ground Truth coordinates. These metrics are calculated based on data collected in $\texttt{E}_{113}$ experiment, **considering the area of the experiment, which is 8.58 meters long**.

Accuracy (Mean Average Error)

The **Mean Absolute Error (MAE)**, in the tables Table 7.1 and Table 7.3, indicate that the coordinate estimations in the S8 - ExpEnv4 have an error of approximately 0.16 m in the x-coordinate and 0.14 m in the y-coordinate (these values are taken as upper limit of the **MAE** from both tables). Fortunately, these errors can be reduced when considering a smaller area of observation, specifically within a 10.08-meter distance from the anchor baseline, as shown in Table 7.5 and Table 7.6. The reason is that at larger distances, in the S8 - ExpEnv4 environment, there is a strong signal interference from an electrical enclose, which

Method	Coordinate	MAE	MSE
Error(C_{UWB}(E_{118}, Tag 1), $RS_{0.5m}$(Tag 1))	x_{UWB}	0.1444	0.0306
	y_{UWB}	0.1075	0.0232
Error(C_{UWB}(E_{118}, Tag 2), $RS_{0.5m}$(Tag 2))	x_{UWB}	0.1631	0.0404
	y_{UWB}	0.1366	0.0471
Error(C_{UWB}(E_{118}, Tag 3), $RS_{0.5m}$(Tag 3))	x_{UWB}	0.1337	0.0327
	y_{UWB}	0.1188	0.0245
Error(C_{P2R}(E_{118}, Tag 1), $RS_{0.5m}$(Tag 1))	x_{P2R}	0.1009	0.016
	y_{P2R}	0.2343	0.1157
Error(C_{P2R}(E_{118}, Tag 2), $RS_{0.5m}$(Tag 2))	x_{P2R}	0.0924	0.0145
	y_{P2R}	0.201	0.0743
Error(C_{P2R}(E_{118}, Tag 3), $RS_{0.5m}$(Tag 3))	x_{P2R}	0.1302	0.0259
	y_{P2R}	0.3279	0.26
Error(C_{Opt}(E_{118}, Tag 1), $RS_{0.5m}$(Tag 1))	x_{Opt}	0.3194	0.1193
	y_{Opt}	0.1509	0.0497
Error(C_{Opt}(E_{118}, Tag 2), $RS_{0.5m}$(Tag 2))	x_{Opt}	0.3163	0.1225
	y_{Opt}	0.12	0.0226
Error(C_{Opt}(E_{118}, Tag 3), $RS_{0.5m}$(Tag 3))	x_{Opt}	0.2689	0.0855
	y_{Opt}	0.5384	0.8775

Table 7.3 Comparative evaluation metrics showing the discrepancy between estimated and reference Ground Truth coordinates. These metrics are calculated based on data collected in E_{118} experiment, **considering the entire area of the experiment, which is 17.08 meters long**.

is discussed in Section 3.4.3. The errors in estimations are even smaller in the data collected in S301 - ExpEnv3 environment. This suggests that the localization system is relatively accurate, especially in the y coordinate, and confirms that UWB localization **can achieve** 10 cm accuracy under ideal conditions [31] and shorter ranges (within 10.08 meters).

In general, the UWB coordinates show the lowest **Mean Absolute Error (MAE)** values for both x and y coordinates, considering the entire area of observation within 17.08-meter distance from anchor baseline. However, based on analysis performed on data collected during E_{118} experiment, **Pixel-to-Real method outperforms** even the **UWB** method along x-coordinate, as shown in Table 7.3. This is due to interference caused by the electrical enclosure, which affects the UWB estimations. The advantage of the Pixel-to-Real model is that it uses only video images as an input and, therefore, is not exposed to signal interference. The fact that Pixel-to-Real method outperforms the UWB method is further demonstrated in the Table 7.6. This table contains statistics based on data collected within a 10.08-meter distance from anchor baseline, excluding the area, where the electrical enclosure is located. It shows that the UWB method again outperforms the Pixel-to-Real method when the area of observation is limited to a smaller area.

Overall, the **Pixel-to-Real** method performs **better** than **Optical** method, but produces less accurate coordinates compared to UWB coordinates.

Consistency (Mean Squared Error)

Method	Coordinate	MAE	MSE
Error($C_{UWB}(E_{124}$, Tag 1), $RS_{0.5m}$(Tag 1))	x_{UWB}	0.0568	0.0049
	y_{UWB}	0.057	0.0041
Error($C_{UWB}(E_{124}$, Tag 2), $RS_{0.5m}$(Tag 2))	x_{UWB}	0.0341	0.0016
	y_{UWB}	0.0465	0.0027
Error($C_{P2R}(E_{124}$, Tag 1), $RS_{0.5m}$(Tag 1))	x_{P2R}	0.0821	0.0082
	y_{P2R}	0.0681	0.0103
Error($C_{P2R}(E_{124}$, Tag 2), $RS_{0.5m}$(Tag 2))	x_{P2R}	0.0998	0.0116
	y_{P2R}	0.0393	0.0023
Error($C_{Opt}(E_{124}$, Tag 1), $RS_{0.5m}$(Tag 1))	x_{Opt}	0.1787	0.0354
	y_{Opt}	0.1368	0.0289
Error($C_{Opt}(E_{124}$, Tag 2), $RS_{0.5m}$(Tag 2))	x_{Opt}	0.1256	0.018
	y_{Opt}	0.0746	0.0099

Table 7.4 Comparative evaluation metrics showing the discrepancy between estimated and reference Ground Truth coordinates. These metrics are calculated based on data collected in E_{124} experiment, **considering the area of the experiment, which is 8.08 meters long**.

Method	Coordinate	MAE	MSE
Error($C_{UWB}(E_{109}$, Tag 1), $RS_{0.5m}$(Tag 1))	x_{UWB}	0.0445	0.003
	y_{UWB}	0.0218	0.001
Error($C_{P2R}(E_{109}$, Tag 1), $RS_{0.5m}$(Tag 1))	x_{P2R}	0.054	0.0054
	y_{P2R}	0.0666	0.0123
Error($C_{Opt}(E_{109}$, Tag 1), $RS_{0.5m}$(Tag 1))	x_{Opt}	0.2158	0.0552
	y_{Opt}	0.1404	0.0348

Table 7.5 Comparative evaluation metrics showing the discrepancy between estimated and reference Ground Truth coordinates. These metrics are calculated based on data collected in E_{109} experiment, **considering the reduced area of the experiment, which is 10.08 meters long**.

The **Mean Squared Error (MSE)** for UWB is the lowest, showing that the UWB method is more consistent with less error variance. On the other hand, the Optical method has the highest **MSE** values, indicating higher error variance. Finally, the pixel-to-real method has moderate **MSE** values, better than optical but worse than UWB.

Overall, Pixel-to-Real method performs well. It offers a good balance between accuracy and consistency. It has lower accuracy compared to UWB coordinates, but performs better than optical method. This method could even outperform the UWB method in the complex environments causing the strong signal interference.

Furthermore, the reduction in range significantly benefits all methods, particularly the UWB method.

Visual analysis with respect to the Ground Truth coordinates

To further evaluate the accuracy of the estimated UWB coordinates, as well as coordinates provided by Optical and Pixel-to-Real methods, we can visualize

Method	Coordinate	MAE	MSE
Error(C_{UWB}(E_{118}, Tag 1), $RS_{0.5m}$(Tag 1))	$\mathtt{x_{UWB}}$	0.1095	0.0185
	$\mathtt{y_{UWB}}$	0.109	0.0145
Error(C_{UWB}(E_{118}, Tag 2), $RS_{0.5m}$(Tag 2))	$\mathtt{x_{UWB}}$	0.1066	0.0165
	$\mathtt{y_{UWB}}$	0.1108	0.0361
Error(C_{UWB}(E_{118}, Tag 3), $RS_{0.5m}$(Tag 3))	$\mathtt{x_{UWB}}$	0.0881	0.0152
	$\mathtt{y_{UWB}}$	0.095	0.0101
Error(C_{P2R}(E_{118}, Tag 1), $RS_{0.5m}$(Tag 1))	$\mathtt{x_{P2R}}$	0.0796	0.0107
	$\mathtt{y_{P2R}}$	0.075	0.0122
Error(C_{P2R}(E_{118}, Tag 2), $RS_{0.5m}$(Tag 2))	$\mathtt{x_{P2R}}$	0.087	0.0094
	$\mathtt{y_{P2R}}$	0.1616	0.0533
Error(C_{P2R}(E_{118}, Tag 3), $RS_{0.5m}$(Tag 3))	$\mathtt{x_{P2R}}$	0.0974	0.0176
	$\mathtt{y_{P2R}}$	0.1095	0.0239
Error(C_{Opt}(E_{118}, Tag 1), $RS_{0.5m}$(Tag 1))	$\mathtt{x_{Opt}}$	0.2218	0.0561
	$\mathtt{y_{Opt}}$	0.0671	0.0058
Error(C_{Opt}(E_{118}, Tag 2), $RS_{0.5m}$(Tag 2))	$\mathtt{x_{Opt}}$	0.2012	0.0464
	$\mathtt{y_{Opt}}$	0.114	0.0194
Error(C_{Opt}(E_{118}, Tag 3), $RS_{0.5m}$(Tag 3))	$\mathtt{x_{Opt}}$	0.195	0.0466
	$\mathtt{y_{Opt}}$	0.1435	0.0318

Table 7.6 Comparative evaluation metrics showing the discrepancy between estimated and reference Ground Truth coordinates. These metrics are calculated based on data collected in $\mathtt{E_{118}}$ experiment, **considering the reduced area of the experiment, which is 10.08 meters long**.

the collected data.

It is important to note that throughout our analysis we have created a variety of diagrams. Not all of these diagrams are attached in Appendix A. However, all of them are available on GitHub[3]. In the following text we will provide only reference path (in the footnotes), unless stated otherwise.

Scatter plots Figure A.3 – Figure A.13 illustrate the correlation between the reference Ground Truth coordinates and UWB, Pixel-to-Real and Optical coordinates collected in $\mathtt{E_{109}}$, $\mathtt{E_{113}}$, $\mathtt{E_{118}}$ and $\mathtt{E_{124}}$ experiments. Figure A.3 and Figure A.6 reveal that the estimated UWB coordinates closely match the Ground Truth coordinates in the area up to the 10.08 meters from the camera. However, beyond this distance, the UWB coordinates become more scattered and less accurate, particularly in S8 - ExpEnv4. This also confirms the observations described using statistical metrics showing the accuracy.

Figure A.4, Figure A.7, Figure A.9 and Figure A.12 are interesting for us, as they highlight the performance of the Pixel-to-Real model in different environments. Let us remember that the Pixel-to-Real model was trained on the data collected during $\mathtt{E_{109}}$ experiment.

Comparing the Figure A.3, showing the estimated UWB coordinates, and Figure A.4, showing the coordinates predicted by Pixel-to-Real model, we can state that the Pixel-to-Real model performs well in the S8 - ExpEnv4. On the

[3]https://github.com/Razyapoo/Master-Thesis/tree/main/PixelToReal, Optical and UWB evaluation/Relusts of evaluation (Plots, Statistics)/Plots/Comparison with ground truth coordinates

other hand, Figure A.7 demonstrates the model's performance in the new, E_{113} environment.

Figure A.10 (this plot is intended only for Person 1, for other people please find plots under the name of the experiment) indicates that the Pixel-to-Real model slightly **outperforms** the UWB method, especially on distances larger than 10.08 meters from the anchor baseline and within the area of the electrical enclosure. However, considering the results from other experiments, within the reduced area of a 10.08-meters distance from the anchor baseline, the UWB coordinates are more precised than the coordinates predicted by Pixel-to-Real method.

Exploring Figure A.5, Figure A.8, Figure A.11 and Figure A.13, we can conclude that the Pixel-to-Real method shows higher accuracy in estimated coordinates compared to Optical method, which again confirm the accuracy of the Pixel-to-Real method. Other scatter plots showing the evaluation of UWB, Pixel-to-Real and Optical methods compared to reference Ground Truth coordinates are available here [4].

Box plots and histograms Figure A.14 – Figure A.17 illustrate the discrepancy between the estimated x and y coordinates and the corresponding reference Ground Truth coordinates in both environments. These diagrams reveal that the UWB method performs similarly in both, S301 - ExpEnv3 and S8 - ExpEnv4 environments, as does the optical method. However, in S301 - ExpEnv3, the Pixel-to-Real model estimates the y-coordinate less accurate compared to its estimations in S8 - ExpEnv4. Conversely, the x-coordinate is predicted with approximately the same accuracy in both environments. This is further illustrated in Figure A.14 and Figure A.16, and Figure A.15 and Figure A.17. An interesting point is that the x-coordinate estimated using the UWB method in E_{109} shows an error of 1 meter. This corresponds to the coordinates (position) estimation shown in Figure A.3 in 6th row, counting from left. Figure A.3 shows that this position is located in the area close to the electrical enclosure.

Other descriptive plots [5] highlight the distribution of errors in collected coordinates over time. The largest errors are observed at increased distances from the origin, likely due to the greater distance and potential interference effects.

Error as the Euclidean distance from Ground Truth coordinates

This analysis is aimed at calculating the error as the distance between estimated people's positions and corresponding reference positions across E_{109}, E_{113}, E_{118} and E_{124} experiments. In the following section Section 7.5, we will extend this analysis by calculating the distances between people as estimated by each method, including UWB, Optical, and Pixel-to-Real, and then compare these distances with the corresponding distances between people in the Ground Truth reference system.

To accurately evaluate the positions estimated by each method, we calculate the Euclidean distance between the estimated people's positions and reference Ground Truth people's positions for each data point. This provides a clear metric of how closely each method's estimates match the actual reference positions.

The calculated summary statistics, which show the discrepancy between the

[4]Reference vs estimated scatter plots
[5]Error trend over time (in coordinates)

Method	*Mean*	*Median*	*Max*	*Min*	*StdDev*
UWB	0.1069	0.0703	1.0076	0.0036	0.1278
Pixel-to-Real	0.1468	0.1014	0.8315	0.007	0.1362
Optical	0.4265	0.4245	1.4044	0.0559	0.2150

Table 7.7 Summary statistics for distance errors calculated as the Euclidean distance between estimated and reference people's positions (pseudocode: $\|p_{\{\text{method}\}} - p_{\text{ref}}\|$). The data is collected in $\texttt{E}_{109}$, **considering the entire area of the experiment, which is 17.08 meters long**.

Method	*Mean*	*Median*	*Max*	*Min*	*StdDev*
UWB	0.0229	0.0185	0.0791	0.0022	0.0161
Pixel-to-Real	0.2052	0.1453	0.658	0.0099	0.1608
Optical	0.2843	0.2688	0.5576	0.0945	0.1106

Table 7.8 Summary statistics for distance errors calculated as the Euclidean distance between estimated and reference people's positions. The data is collected in $\texttt{E}_{113}$, **considering the entire area of the experiment, which is 8.58 meters long**.

reference and estimated people positions as the Euclidean distance between them is shown in the following tables:

- Table 7.7 for the $\texttt{E}_{109}\texttt{(DA\_S8\_S}_5\texttt{(T1\_A2\_TP}_\texttt{h}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{pn}\texttt{))}$ experiment, considering the entire area of the environment, which is 17.08 meters long.
- Table 7.8 for the $\texttt{E}_{113}\texttt{(DA\_S301\_S}_5\texttt{(T1\_A2\_TP}_\texttt{h}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{pn}\texttt{))}$ experiment. In this experiment we consider only the entire range of the experiment, which is 8.58 meters long.
- Table 7.9 for the $\texttt{E}_{118}\texttt{(DA\_S8\_S}_6\texttt{(T3\_A4\_TP}_\texttt{h}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{p}\texttt{))}$ experiment, considering the entire area of the environment, which is 17.08 meters long.
- Table 7.10 for the $\texttt{E}_{124}\texttt{(DA\_S301\_S}_6\texttt{(T2\_A4\_TP}_\texttt{h}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{p}\texttt{))}$ experiment. In this experiment we consider only the entire range of the experiment, which is 8.08 meters long.
- Table 7.11 for the $\texttt{E}_{109}\texttt{(DA\_S8\_S}_5\texttt{(T1\_A2\_TP}_\texttt{h}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{pn}\texttt{))}$ experiment, considering the reduced area of the environment, which is 10.08 meters long.
- Table 7.12 for the $\texttt{E}_{118}\texttt{(DA\_S8\_S}_6\texttt{(T3\_A4\_TP}_\texttt{h}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{p}\texttt{))}$ experiment, considering the reduced area of the environment, which is 10.08 meters long.

All the data presented in the tables is available in the main repository of the thesis on GitHub [6]. This repository also contains data about other participants, where applicable.

Mean Euclidean distance error

The **mean euclidean distance error** indicates that, on average, when considering the reduced area of the experiment within the 10.08-meters distance

[6]Distance error statistics between estimated and reference positions

Method	*Mean*	*Median*	*Max*	*Min*	*StdDev*
UWB	0.2035	0.1936	0.5889	0.0508	0.1136
Pixel-to-Real	0.2753	0.1797	0.8195	0.0208	0.241
Optical	0.3698	0.3482	0.8193	0.0717	0.1831

Table 7.9 Summary statistics for distance errors calculated as the Euclidean distance between estimated and reference people's positions. The data is collected in $\mathtt{E}_{118}$, **considering the entire area of the experiment, which is 17.08 meters long**. These statistics correspond only to first participant of the experiment, wearing Tag 1.

Method	*Mean*	*Median*	*Max*	*Min*	*StdDev*
UWB	0.0881	0.0724	0.1594	0.0518	0.0379
Pixel-to-Real	0.1138	0.101	0.2813	0.0255	0.0787
Optical	0.2473	0.2488	0.3344	0.1623	0.0594

Table 7.10 Summary statistics for distance errors calculated as the Euclidean distance between estimated and reference people's positions. The data is collected in $\mathtt{E}_{124}$, **considering the entire area of the experiment, which is 8.08 meters long**. These statistics correspond only to first participant of the experiment, wearing Tag 1.

Method	*Mean*	*Median*	*Max*	*Min*	*StdDev*
UWB	0.0536	0.0548	0.1704	0.0051	0.0327
Pixel-to-Real	0.0961	0.067	0.6228	0.007	0.0922
Optical	0.2743	0.2607	0.777	0.0559	0.1223

Table 7.11 Summary statistics for distance errors calculated as the Euclidean distance between estimated and reference people's positions. The data is collected in $\mathtt{E}_{109}$, **considering the reduced area of the experiment, which is 10.08 meters long**.

Method	*Mean*	*Median*	*Max*	*Min*	*StdDev*
UWB	0.1719	0.1665	0.3098	0.0978	0.0608
Pixel-to-Real	0.1268	0.1057	0.2816	0.0208	0.0854
Optical	0.2356	0.2086	0.3757	0.0717	0.0831

Table 7.12 Summary statistics for distance errors calculated as the Euclidean distance between estimated and reference people's positions. The data is collected in $\mathtt{E}_{118}$, **considering the reduced area of the experiment, which is 10.08 meters long**. These statistics correspond only to first participant of the experiment, wearing Tag 1.

from anchor baseline, the estimated UWB people's positions are 0.1 meter away from the reference people's positions. However, for larger areas of the experiments, the distance error increases. This can be seen particularly in Table 7.9.

Comparing all summaries from Table 7.7 to Table 7.12, we can conclude that, on average, the Pixel-to-Real method estimates people's positions with an approximately equal distance error in each environment. The exception, however, is the data collected in the $\mathtt{E}_{\mathtt{118}}$ experiment, considering the entire area of the experiment, which includes an area 17.08 meters long from the anchor baseline. The summary statistics for this situation are shown in table Table 7.9. Here, the mean distance error is 0.27, which is relatively high. This is due to the 17.08 meters long area of the experiment, which affects the Pixel-to-Real model, as people are detected poorly at larger distances from a camera. It is important to note that the Pixel-to-Real method assumes the pixel coordinates of the center of the bottom edge of the bounding box of the detected person.

The Optical method has the highest mean distance error in each experiment, regardless of whether the entire or reduced area of the experiment environment is considered. This is mainly due to the error along the x-coordinate, which is discussed in detail in the previous analysis, at the beginning of Section 7.4.1. This also suggests, that the Pixel-to-Real method outperforms the Optical method, especially at larger distances from a camera.

Error as Euclidean Distance: Median, Min, Max, and Std Dev

The **median distance error** across all methods and experiments are lower than the mean errors, indicating a skew towards smaller errors with the presence of significant outliers. The UWB method has the least skew, while the Optical method has the most.

Across all experiments, UWB consistently shows the **lowest** minimum (**Min**), maximum (**Max**) and standard deviation of errors, indicating fewer extreme deviations. The Optical method often has the highest errors, indicating greater variability and less predictable performance, especially at larger distances from the camera. Pixel-to-Real method performs better than the Optical method, but generally worse than the UWB method. This method shows better performance in an area closer to the camera, but on the other hand worse at long distances.

Furthermore, histogram plots shown in Figure A.18, Figure A.19, Figure A.20 and Figure A.21 and scatter plots[7], show the errors in position estimations as the Euclidean distance between the estimated and reference positions. These diagrams reveal that UWB and Pixel-to-Real methods estimate people's positions relatively close to the corresponding reference positions. However, in $\mathtt{E}_{\mathtt{109}}$, there is an outlier, which has 1 meter distance error. This outlier corresponds to the position located within the area close to the electrical enclosure, as was revealed before. On the other hand, distance errors for the Optical method are higher compared to UWB and Pixel-to-Real methods.

[7]Distance errors between estimated and reference positions

7.4.2 Evaluation of Pixel-to-Real coordinates based on UWB coordinates

In this section, we will evaluate the coordinates obtained using the Pixel-to-Real method.

By comparing the coordinates estimated using the Pixel-to-Real model with UWB coordinates, we can evaluate the Pixel-to-Real model and show the error in the coordinates' estimation relative to the UWB coordinates.

The main focus of this section is to evaluate the coordinates obtained using the Pixel-to-Real model by comparing them to the UWB coordinates, as the Pixel-to-Real model is trained on the UWB measurements. However, we have also compared the coordinates provided by the Optical method to those provided by the UWB method, for informational purposes only. Although, the results of this comparison are available on the GitHub [8], this analysis is not described in the following text.

Previous analysis has demonstrated that the UWB method performs well, consistently showing low error metrics across various experiments and environments, especially in smaller areas of around 10.08 meters long. In addition, the use of UWB coordinates as a reference system facilitates the deployment of our indoor positioning system in the customer's premises, as the complexity of creating a reference system consisting of paper stickers strongly depends on the type of an environment. As a result, creating such helping system can be very difficult and time-consuming process.

These factors allow us to use the UWB system as a reference system during the calibration process in the deployment stage.

Similar to the previous analysis, we will use the following notation to express the error in coordinates estimation for Pixel-to-Real method:

$$\mathtt{Error(C_{P2R},\ C_{UWB})} = \begin{cases} |\mathtt{x_{P2R}} - \mathtt{x_{UWB}}|, & \text{for } \mathtt{x} \text{ coordinate} \\ |\mathtt{y_{P2R}} - \mathtt{y_{UWB}}|, & \text{for } \mathtt{y} \text{ coordinate} \end{cases} \tag{7.2}$$

A pair $(\mathtt{x_{UWB}},\ \mathtt{y_{UWB}})$ represents the coordinates obtained from the UWB method; a pair $(\mathtt{x_{P2R}},\ \mathtt{y_{P2R}})$ represents the coordinates obtained from the Pixel-to-Real method.

As the UWB technology is used only during the deployment stage, we selected to analyze the $\mathtt{E_{109}}$ and $\mathtt{E_{113}}$ experiments, because the closely emulate the deployment stage. These experiments produce lower errors in UWB estimations compared to the $\mathtt{E_{118}}$ and $\mathtt{E_{124}}$ experiments.

Although the UWB-based evaluation of the data collected during the $\mathtt{E_{118}}$ and $\mathtt{E_{124}}$ experiments is not shown in this section, this analysis is available in the main repository of the thesis on GitHub[9], in the corresponding folders (in each type of diagrams).

The calculated error metrics are provided in:

[8]https://github.com/Razyapoo/Master-Thesis/tree/main/PixelToReal, Optical and UWB evaluation/Relusts of evaluation (Plots, Statistics)/Statistics/Comparison with uwb coordinates/Error statistics for each coordinate (MAE, MSE and RMSE)

[9]https://github.com/Razyapoo/Master-Thesis/tree/main/PixelToReal, Optical and UWB evaluation/Relusts of evaluation (Plots, Statistics)/Plots/Comparison with uwb coordinates/

- Table 7.13 for the $\mathtt{E_{109}(DA\_S8\_S_5(T1\_A2\_TP_h\_M_d\_W_{pn}))}$ experiment, considering the entire area of the environment, which is 17.08 meters long.
- Table 7.14 for the $\mathtt{E_{109}(DA\_S8\_S_5(T1\_A2\_TP_h\_M_d\_W_{pn}))}$ experiment, considering the reduced area of the environment, which is 10.08 meters long.
- Table 7.15 for the $\mathtt{E_{113}(DA\_S301\_S_5(T1\_A2\_TP_h\_M_d\_W_{pn}))}$ experiment.

Method	Coordinate	*MAE*	*MSE*
Error(C_{P2R}(E_{109}, Tag 1), C_{UWB}(E_{109}, Tag 1))	$\mathtt{x_{P2R}}$	0.0510	0.0071
	$\mathtt{y_{P2R}}$	0.0814	0.0164

Table 7.13 Comparative evaluation metrics showing the discrepancy between estimated and UWB coordinates. These metrics are calculated based on data collected in $\mathtt{E_{109}}$ experiment, **considering the entire area of the experiment, which is 17.08 meters long**.

Method	Coordinate	*MAE*	*MSE*
Error(C_{P2R}(E_{109}, Tag 1), C_{UWB}(E_{109}, Tag 1))	$\mathtt{x_{P2R}}$	0.0323	0.0034
	$\mathtt{y_{P2R}}$	0.0597	0.0113

Table 7.14 Comparative evaluation metrics showing the discrepancy between estimated and UWB coordinates. These metrics are calculated based on data collected in $\mathtt{E_{109}}$ experiment, **considering the reduced area of the experiment, which is 10.08 meters long**.

Method	Coordinate	*MAE*	*MSE*
Error(C_{P2R}(E_{113}, Tag 1), C_{UWB}(E_{113}, Tag 1))	$\mathtt{x_{P2R}}$	0.0759	0.0125
	$\mathtt{y_{P2R}}$	0.1664	0.0551

Table 7.15 Comparative evaluation metrics showing the discrepancy between estimated and UWB coordinates. These metrics are calculated based on data collected in $\mathtt{E_{113}}$ experiment, **considering the entire area of the experiment, which is 8.58 meters long**.

Accuracy (Mean Absolute Error)

Let's explore the Table 7.13 and Table 7.14 tables, which show the evaluation of the $\mathtt{C_{P2R}(E_{109})}$ coordinates collected in the full and reduced areas of the S8 - ExpEnv4 environment, respectively, during the $\mathtt{E_{109}}$ experiment.

We can observe that in the reduced area of the environment, which is 10.08 meters long, the Pixel-to-Real model estimates the coordinates more accurately. Specifically, considering the reduced area, the MAE value for the x-coordinate is 0.0323 meters and for the y-coordinate is 0.0597 meters. In comparison, in the full area, the MAE value for the x-coordinate is 0.0510 meters and for the y-coordinate is 0.0814 meters. This highlights that, on average, the Pixel-to-Real model estimates the coordinates more accurately in the reduced area. It is important to note that the Pixel-to-Real model was trained on the $\mathtt{C_{UWB}(E_{109})}$ coordinates, collected in the S8 - ExpEnv4 environment, during the $\mathtt{E_{109}}$ experiment.

Furthermore, examining the Table 7.15, we can state that the Pixel-to-Real model has worse accuracy in the new S8 - ExpEnv4 environment. The MAE value for the x-coordinate is 0.0759 meters and for the y-coordinate is 0.1664 meters.

Eventually, this comparison shows that the Pixel-to-Real model has good accuracy when compared with UWB. This is important as the Pixel-to-Real model is trained on the UWB coordinates. This implies that it is crucial to have UWB coordinates as precise as possible in relation to the reference ground truth coordinates provided by paper stickers. Moreover, the factor of whether the model was trained and tested in the same environment, in which the coordinates used for the Pixel-to-Real model training were collected, also influences the quality of the coordinates estimated by the model.

The box plots shown in Figure A.22, Figure A.23 and Figure A.24 visually illustrate the observations discovered in Table 7.13, Table 7.14 and Table 7.15.

Error as the Euclidean distance from UWB coordinates

Similar to the analysis performed in Section 7.4.1, we can evaluate the Pixel-to-Real estimations by calculating the Euclidean distance between the estimated Pixel-to-Real positions and the positions provided by the UWB method.

This analysis is very important, as it provides a clear metric of how closely the positions estimated by Pixel-to-Real model matches the positions estimated by UWB method, and allows to evaluate the trained Pixel-to-Real model.

The calculated summary statistics, which show the discrepancy between the UWB positions and positions estimated using the Pixel-to-Real method, are shown in the following tables:

- Table 7.16 for the $\texttt{E}_{\texttt{109}}\texttt{(DA\_S8\_S}_{\texttt{5}}\texttt{(T1\_A2\_TP}_{\texttt{h}}\texttt{\_M}_{\texttt{d}}\texttt{\_W}_{\texttt{pn}}\texttt{))}$ experiment, considering the entire area of the environment, which is 17.08 meters long.
- Table 7.17 for the $\texttt{E}_{\texttt{109}}\texttt{(DA\_S8\_S}_{\texttt{5}}\texttt{(T1\_A2\_TP}_{\texttt{h}}\texttt{\_M}_{\texttt{d}}\texttt{\_W}_{\texttt{pn}}\texttt{))}$ experiment, considering the reduced area of the environment, which is 10.08 meters long.
- Table 7.18 for the $\texttt{E}_{\texttt{113}}\texttt{(DA\_S301\_S}_{\texttt{5}}\texttt{(T1\_A2\_TP}_{\texttt{h}}\texttt{\_M}_{\texttt{d}}\texttt{\_W}_{\texttt{pn}}\texttt{))}$ experiment. In this experiment we consider only the entire range of the experiment, which is 8.58 meters long.

Method	*Mean*	*Median*	*Max*	*Min*	*StdDev*
Pixel-to-Real	0.1041	0.0689	0.8210	0.0023	0.1128

Table 7.16 Summary statistics for distance errors calculated as the Euclidean distance between estimated (Pixel-to-Real) and reference (UWB) people's positions (pseudo-code: $\|p_{\text{P2R}} - p_{\text{UWB}}\|$). The data is collected in $\texttt{E}_{\texttt{109}}$, **considering the entire area of the experiment, which is 17.08 meters long**.

The **mean** Euclidean distance error, from Table 7.16, Table 7.17 and Table 7.18, shows that the Pixel-to-Real model estimates the positions better in the environment where the model was trained (S301 - ExpEnv3) compared to the environment new to the model (S8 - ExpEnv4). The mean value is 0.0750 meters in the reduced area and 0.1041 meters in the full area of the $\texttt{E}_{\texttt{109}}$ experiment,

Method	*Mean*	*Median*	*Max*	*Min*	*StdDev*
Pixel-to-Real	0.0750	0.0450	0.6025	0.0023	0.0956

Table 7.17 Summary statistics for distance errors calculated as the Euclidean distance between estimated (Pixel-to-Real) and reference (UWB) people's positions. The data is collected in `E109`, **considering the reduced area of the experiment, which is 10.08 meters long**.

Method	*Mean*	*Median*	*Max*	*Min*	*StdDev*
Pixel-to-Real	0.2058	0.1423	0.6698	0.0128	0.1601

Table 7.18 Summary statistics for distance errors calculated as the Euclidean distance between estimated (Pixel-to-Real) and reference (UWB) people's positions. The data is collected in `E113`, **considering the entire area of the experiment, which is 8.58 meters long**.

while in the full area of the `E113` experiment the mean value is 0.2058. This indicates that the Pixel-to-Real model's performance **degrades** when applied to environments different from those it was trained on, highlighting the importance of environmental consistency during model training and its subsequent use.

It is important to note that the maximum (**Max**) distance error in the S8 - ExpEnv4 environment (Table 7.16 and Table 7.17), where `E109` is performed and the model is trained, is larger than in the S301 - ExpEnv3 environment (Table 7.18), where the `E113` is performed. The maximum value is 0.6025 meters in the reduced area and 0.8210 meters in the full area of the `E109` experiment, while in the `E113` the maximum value is 0.6698 meters. This discrepancy is likely caused by the signal interference observed in the area next to the electrical enclosure, which is located at the end of the S8 - ExpEnv4 environment, as discussed in Section 3.4.3.

Figure A.25, Figure A.26 and Figure A.27 show histograms that depict the distribution of Euclidean distance errors in the data (positions) from the `E109` experiment (full area of 17.08 meters and reduced area of 10.08 meters) and the `E113` experiment (full area of 8.58 meters). These histograms reveal that, although the Pixel-to-Real model records the highest (**Max**) error in the S8 - ExpEnv4 environment during `E109`, the majority of distance errors are near zero when compared to the S301 - ExpEnv3 environment, where the Pixel-to-Real model demonstrates larger distance errors.

7.5 Evaluation of distances between people

An accurate estimation of the distance between individuals is crucial for maintaining safe social distancing and preventing hazardous situations. These conditions had to be satisfied during the Covid-19 pandemic. Additionally, by placing anchors on hazardous equipment, we can notify people in time when someone is getting very close to dangerous areas.

To achieve this, we can use coordinates calculated by various methods such as UWB, Pixel-to-Real, and Optical. However, as was shown during the previous analysis in Section 7.4, the precision of these coordinates may vary between methods. Therefore, it becomes important to evaluate not just the coordinates themselves, but also the distances between tags (people).

It is important to note that even if the estimated coordinates are inaccurate, the distances between people might still be correctly estimated. This can occur because any consistent shift (compared to Ground Truth) in people's coordinates can be canceled out when calculating distances. Hence, evaluating distances between individuals can provide an additional metric for the evaluation of UWB, Pixel-to-Real and Optical methods.

During this evaluation, we will perform the comparison on the data collected during the following experiments:

- $\texttt{E}_{118}$`(DA_S8_`$\texttt{S}_6$`(T3_A4_`$\texttt{TP}_\texttt{h}$`_`$\texttt{M}_\texttt{d}$`_`$\texttt{W}_\texttt{p}$`))`
- $\texttt{E}_{124}$`(DA_S301_`$\texttt{S}_6$`(T2_A4_`$\texttt{TP}_\texttt{h}$`_`$\texttt{M}_\texttt{d}$`_`$\texttt{W}_\texttt{p}$`))`

We consider only $\texttt{E}_{118}$ and $\texttt{E}_{124}$ experiments, because they involve at least two participants. Moreover, both experiments follow the $\texttt{S}_6$ scenario, where people are walking parallel, along the predefined lines (paper stickers), following the $\texttt{W}_\texttt{p}$ walking pattern. This allows to easily check the distance between people.

7.5.1 Evaluation based on Ground Truth distances between people

In this section, we will perform the evaluation of the distances between people obtained using UWB, Pixel-to-Real and Optical methods and compare them to distances between people calculated in the reference Ground Truth system $\texttt{RS}_{\texttt{0.5m}}$.

To show the discrepancy between distances, we will use the following notation:

$$\texttt{Error(DPeople}_{\texttt{\{method\}}}\texttt{(tagid\_1, tagid\_2), DPeople}_{\texttt{RS}}\texttt{(tagid\_1, tagid\_2))} \tag{7.3}$$

where $\texttt{DPeople}_{\texttt{\{method\}}}$ `(tagid_1, tagid_2)` is a function returning the distance between two people, which is calculated using the coordinates provided by `method`. The `method` variable can take on the following values: `UWB, P2R, Opt`.

The calculated error metrics are provided in:

- Table 7.19 for the $\texttt{E}_{118}$`(DA_S8_`$\texttt{S}_6$`(T3_A4_`$\texttt{TP}_\texttt{h}$`_`$\texttt{M}_\texttt{d}$`_`$\texttt{W}_\texttt{p}$`))` experiment, considering the entire area of the environment, which is 17.08 meters long
- Table 7.20 for the $\texttt{E}_{118}$`(DA_S8_`$\texttt{S}_6$`(T3_A4_`$\texttt{TP}_\texttt{h}$`_`$\texttt{M}_\texttt{d}$`_`$\texttt{W}_\texttt{p}$`))` experiment, considering the reduced area of the environment, which is 10.08 meters long
- Table 7.21 for the $\texttt{E}_{124}$`(DA_S301_`$\texttt{S}_6$`(T2_A4_`$\texttt{TP}_\texttt{h}$`_`$\texttt{M}_\texttt{d}$`_`$\texttt{W}_\texttt{p}$`))` experiment. In this experiment we consider only the entire range of the experiment, which is 8.08 meters long

In order to evaluate the calculated distances, we can consider several different metrics: MAE, Median, Min and Max.

Accuracy (Mean Absolute Error)

Method	MAE	$Median$	Min	Max
Error(DPeople$_{\text{UWB}}$(Tag 1, Tag 2), DPeople$_{\text{RS}}$(Tag 1, Tag 2))	0.175	0.070	0.005	0.470
Error(DPeople$_{\text{UWB}}$(Tag 1, Tag 3), DPeople$_{\text{RS}}$(Tag 1, Tag 3))	0.130	0.029	0.011	0.431
Error(DPeople$_{\text{UWB}}$(Tag 2, Tag 3), DPeople$_{\text{RS}}$(Tag 2, Tag 3))	0.145	-0.028	0.000	0.382
Error(DPeople$_{\text{P2R}}$(Tag 1, Tag 2), DPeople$_{\text{RS}}$(Tag 1, Tag 2))	0.143	0.093	0.008	0.532
Error(DPeople$_{\text{P2R}}$(Tag 1, Tag 3), DPeople$_{\text{RS}}$(Tag 1, Tag 3))	0.152	-0.063	0.010	0.470
Error(DPeople$_{\text{P2R}}$(Tag 2, Tag 3), DPeople$_{\text{RS}}$(Tag 2, Tag 3))	0.177	-0.122	0.022	0.445
Error(DPeople$_{\text{Opt}}$(Tag 1, Tag 2), DPeople$_{\text{RS}}$(Tag 1, Tag 2))	0.050	0.005	0.001	0.224
Error(DPeople$_{\text{Opt}}$(Tag 1, Tag 3), DPeople$_{\text{RS}}$(Tag 1, Tag 3))	0.161	0.010	0.004	0.936
Error(DPeople$_{\text{Opt}}$(Tag 2, Tag 3), DPeople$_{\text{RS}}$(Tag 2, Tag 3))	0.191	0.012	0.000	1.510

Table 7.19 Evaluation metrics showing the discrepancy between the reference Ground Truth and estimated distances between people. The **Min** and **Max** values show an absolute value. These metrics are calculated based on data collected in E_{118} experiment, **considering the entire area of the experiment, which is 17.08 meters long**.

Method	MAE	$Median$	Min	Max
Error(DPeople$_{\text{UWB}}$(Tag 1, Tag 2), DPeople$_{\text{RS}}$(Tag 1, Tag 2))	0.151	0.127	0.005	0.370
Error(DPeople$_{\text{UWB}}$(Tag 1, Tag 3), DPeople$_{\text{RS}}$(Tag 1, Tag 3))	0.113	0.043	0.023	0.431
Error(DPeople$_{\text{UWB}}$(Tag 2, Tag 3), DPeople$_{\text{RS}}$(Tag 2, Tag 3))	0.125	-0.104	0.000	0.318
Error(DPeople$_{\text{P2R}}$(Tag 1, Tag 2), DPeople$_{\text{RS}}$(Tag 1, Tag 2))	0.106	0.093	0.008	0.255
Error(DPeople$_{\text{P2R}}$(Tag 1, Tag 3), DPeople$_{\text{RS}}$(Tag 1, Tag 3))	0.127	-0.046	0.010	0.385
Error(DPeople$_{\text{P2R}}$(Tag 2, Tag 3), DPeople$_{\text{RS}}$(Tag 2, Tag 3))	0.167	-0.126	0.022	0.425
Error(DPeople$_{\text{Opt}}$(Tag 1, Tag 2), DPeople$_{\text{RS}}$(Tag 1, Tag 2))	0.026	-0.010	0.001	0.053
Error(DPeople$_{\text{Opt}}$(Tag 1, Tag 3), DPeople$_{\text{RS}}$(Tag 1, Tag 3))	0.033	-0.017	0.004	0.134
Error(DPeople$_{\text{Opt}}$(Tag 2, Tag 3), DPeople$_{\text{RS}}$(Tag 2, Tag 3))	0.021	-0.003	0.000	0.072

Table 7.20 Evaluation metrics showing the discrepancy between the reference Ground Truth and estimated distances between people. The **Min** and **Max** values show an absolute value. These metrics are calculated based on data collected in E_{118} experiment, **considering the reduced area of the experiment, which is 10.08 meters long**.

The **Mean Absolute Error (MAE)** helps to understand on average discrepancy between the estimated distances and reference Ground Truth distances between people. Lower MAE values signify higher accuracy, meaning the estimated distances are closer to the reference Ground Truth distances.

Table 7.19 shows the statistics calculated for the entire area of the E_{118} experiment. It suggests that on average, the MAE for all methods is approximately 0.15 meters. Although, this value decreases when considering a reduced area of 10.08 meters from the anchor baseline, it still remains around 0.12 meters for UWB and Pixel-to-Real methods, as shown in Table 7.20. Interestingly, in this case, the Optical method shows the lowest error, around 0.03 meters.

It is important to note that E_{118} and E_{124} experiments are challenging to analyze as they more closely emulate real world conditions, such as when people are moving fast. This again highlights the importance of the more controlled movements with UWB boards throughout the calibration process during the deployment stage. Controlled movements allow for gathering more data and processing it more easily, significantly improving the performance of the trained Pixel-to-Real model.

Method	MAE	$Median$	Min	Max
Error(DPeople$_{\mathrm{UWB}}$(Tag 1, Tag 2), DPeople$_{\mathrm{RS}}$(Tag 1, Tag 2))	0.050	0.016	0.012	0.124
Error(DPeople$_{\mathrm{P2R}}$(Tag 1, Tag 2), DPeople$_{\mathrm{RS}}$(Tag 1, Tag 2))	0.079	0.099	0.010	0.152
Error(DPeople$_{\mathrm{Opt}}$(Tag 1, Tag 2), DPeople$_{\mathrm{RS}}$(Tag 1, Tag 2))	0.052	-0.041	0.019	0.146

Table 7.21 Evaluation metrics showing the discrepancy between the reference Ground Truth and estimated distances between people. The **Min** and **Max** values show an absolute value. These metrics are calculated based on data collected in $\mathtt{E}_{124}$ experiment, **considering the entire area of the experiment, which is 8.08 meters long**.

Comparing values from Table 7.19 and Table 7.20 to the MAE values from Table 7.21, we can state that our system perform much better within the area of the experiment, when it is up to 8.08 meters long. It is worth it to note that the Pixel-to-Real model in $\mathtt{E}_{124}$ performs worse than UWB and Optical methods. The MAE value of the Pixel-to-Real model in $\mathtt{E}_{124}$ is 0.079 meters, while UWB and Optical methods show 0.05 and 0.052 meters, respectively.

Median

The **median** value indicates the most common distance error. To compute this metric, we calculate the distance error by subtracting the reference Ground Truth distance from the estimated distance using the following formula:

$$\texttt{error = estimatedDistance - referenceDistance} \tag{7.4}$$

The median value can be negative as well as positive, reflecting whether the estimated distances tend to be shorter or longer than the reference Ground Truth distances between people. If the value is negative, then the estimated distance is shorter than the reference one, and vice versa.

Exploring the Table 7.19, Table 7.20 and Table 7.21 tables, we can state that the sign of median values remains consistent in both the UWB and Pixel-to-Real methods. However, the sign of the median value in the Optical method changes from positive to negative. This suggests that with increased distance from the camera, the Optical method starts underestimating the distance between people. This is potentially due to the increased distortion and perspective issues. This assertion is further supported by the box plots shown in Figure A.28, Figure A.29 and Figure A.30. These box plots show the distribution of errors in distances between people estimated by UWB, Pixel-to-Real and Optical methods compared to the Ground Truth distances between people. It is important to note that these plots contains negative and positive distance error values.

Additionally, by comparing Figure A.28 and Figure A.29, we can observe that each method starts underestimating the distance between each pair of people when extending the area of the experiment from 10.08 meters to 17.08 meters.

Min and Max

The minimum (**Min**) and maximum (**Max**) values are calculated using the absolute values of distance errors. In other words, they show a minimum and maximum deviation between the estimated and reference Ground Truth distances.

Comparing Table 7.19 and Table 7.20, we can observe that UWB shows tendency to overestimate distances in larger areas, with reduced errors in smaller areas, as seen in the **Max** distance error value. This trend can also be observed in Figure A.28, Figure A.29 and Figure A.30, when comparing them.

The Pixel-to-Real model also shows improvements in smaller area. The Optical method, compared to UWB and Pixel-to-Real methods, significantly improves the distance estimations, suggesting that it may struggle with increased distortion and perspective issues at larger distances, especially when detecting the person holding the Tag 3. Maximum value is reduced from 1.510 meters to 0.072 meters. Furthermore, in the reduced area, the Optical method shows the lowest MAE values across all pairs (Table 7.20), with low **Max** errors, indicating that it is highly accurate at shorter distances when estimating the distance between people.

Table 7.21 shows that each localization method benefits in the smaller area of 8.08 meters. The **MAE** and **Max** values suggests that all methods have better performance in smaller areas.

Influence of area size on measurement accuracy

The analysis reveals that the UWB method do not benefit much from reducing the area of the $\texttt{E}_{\texttt{118}}$ experiment. All statistics for UWB method remains almost the same comparing full area (17.08 meters) to reduced area (10.08 meters).

Based on the result from previous analysis in Section 7.4, the UWB method requires more controlled conditions throughout the data acquisition during the deployment stage (considering $\texttt{E}_{\texttt{109}}$ as a more controlled experiment, where the participant walks slower).

Let's remember that we assume to use UWB only throughout the calibration process at the deployment stage to train and evaluate the Pixel-to-Real model. On the other hand, even in less controlled experiments, which have smaller areas, the UWB method performs with a good accuracy. For example, this can be observed in $\texttt{E}_{\texttt{124}}$ experiment, which is 8.08 meters long. Here, the UWB shows 0.05 meters as MAE value.

For other methods, like Pixel-to-Real and Optical, the area size of an experiment environment significantly influences measurement accuracy. Longer area, as in the full area of $\texttt{E}_{\texttt{118}}$, lead to higher maximum errors, particularly for the Optical method. The inaccuracies are primarily due to depth perception limitations and field of view constraints. Conversely, smaller areas, as in the reduced area of $\texttt{E}_{\texttt{118}}$ and $\texttt{E}_{\texttt{124}}$, result in lower maximum errors, indicating better performance in smaller environments.

It is interesting to note that the minimum distance error values are close to zero for all methods, indicating that in some cases, these methods estimate the distance between people correctly. This shows the potential for highly accurate estimations under optimal conditions, particularly in smaller environments.

The Pixel-to-Real method even outperforms UWB in some cases. For example, it has lower $\texttt{Error(DPeople}_{\texttt{P2R}}\texttt{(Tag 1, Tag 2), DPeople}_{\texttt{RS}}\texttt{(Tag 1, Tag 2))}$ value in $\texttt{E}_{\texttt{118}}$ experiment. This is because the $\texttt{E}_{\texttt{118}}$ and $\texttt{E}_{\texttt{124}}$ experiments are less controlled, resulting in worse correction of UWB coordinates. On the other hand, the Pixel-to-Real model is trained on data collected under the more controlled $\texttt{E}_{\texttt{109}}$ experiment, in which UWB data was corrected perfectly, and thus resulted with better accuracy than Pixel-to-Real method.

Eventually, these experiments allow to understand the behavior of the localization methods in different environments under more realistic conditions.

7.5.2 Evaluation of Pixel-to-Real distances between people based on UWB distances between people

In this section we will evaluate the distances between people estimated using the Pixel-to-Real method by comparing them to the distances between people estimated using the UWB method. This analysis aims to further evaluate the trained Pixel-to-Real model.

We will use the following notation to express the error in distance estimation for Pixel-to-Real method:

$$\texttt{Error(DPeople}_{\texttt{P2R}}\texttt{(tagid\_1, tagid\_2), DPeople}_{\texttt{UWB}}\texttt{(tagid\_1, tagid\_2))} \quad (7.5)$$

The calculated error metrics are provided in:

- Table 7.22 for the $\texttt{E}_{118}\texttt{(DA\_S8\_S}_6\texttt{(T3\_A4\_TP}_h\texttt{\_M}_d\texttt{\_W}_p\texttt{))}$ experiment, considering the entire area of the environment, which is 17.08 meters long.
- Table 7.23 for the $\texttt{E}_{118}\texttt{(DA\_S8\_S}_6\texttt{(T3\_A4\_TP}_h\texttt{\_M}_d\texttt{\_W}_p\texttt{))}$ experiment, considering the reduced area of the environment, which is 10.08 meters long.
- Table 7.24 for the $\texttt{E}_{124}\texttt{(DA\_S301\_S}_6\texttt{(T2\_A4\_TP}_h\texttt{\_M}_d\texttt{\_W}_p\texttt{))}$ experiment. In this experiment we consider only the entire range of the experiment, which is 8.08 meters long.

Method	*MAE*	*Median*	*Min*	*Max*
Error(DPeople$_{P2R}$(Tag 1, Tag 2), DPeople$_{UWB}$(Tag 1, Tag 2))	0.263	0.080	0.042	0.725
Error(DPeople$_{P2R}$(Tag 1, Tag 3), DPeople$_{UWB}$(Tag 1, Tag 3))	0.232	-0.154	0.004	0.609
Error(DPeople$_{P2R}$(Tag 2, Tag 3), DPeople$_{UWB}$(Tag 2, Tag 3))	0.197	-0.104	0.001	0.826

Table 7.22 Comparative evaluation metrics showing the discrepancy between estimated distances between people using Pixel-to-Real method and the corresponding distances provided by UWB method. The **Min** and **Max** values show an absolute value. These metrics are calculated based on data collected in $\texttt{E}_{118}$ experiment, **considering the entire area of the experiment, which is 17.08 meters long**.

Method	*MAE*	*Median*	*Min*	*Max*
Error(DPeople$_{P2R}$(Tag 1, Tag 2), DPeople$_{UWB}$(Tag 1, Tag 2))	0.187	-0.042	0.042	0.466
Error(DPeople$_{P2R}$(Tag 1, Tag 3), DPeople$_{UWB}$(Tag 1, Tag 3))	0.163	-0.139	0.004	0.481
Error(DPeople$_{P2R}$(Tag 2, Tag 3), DPeople$_{UWB}$(Tag 2, Tag 3))	0.132	-0.018	0.001	0.392

Table 7.23 Comparative evaluation metrics showing the discrepancy between estimated distances between people using Pixel-to-Real method and the corresponding distances provided by UWB method. The **Min** and **Max** values show an absolute value. These metrics are calculated based on data collected in $\texttt{E}_{118}$ experiment, **considering the reduced area of the experiment, which is 10.08 meters long**.

Method	MAE	$Median$	Min	Max
Error(DPeople$_{\mathrm{P2R}}$(Tag 1, Tag 2), DPeople$_{\mathrm{UWB}}$(Tag 1, Tag 2))	0.069	0.028	0.009	0.162

Table 7.24 Comparative evaluation metrics showing the discrepancy between estimated distances between people using Pixel-to-Real method and the corresponding distances provided by UWB method. The **Min** and **Max** values show an absolute value. These metrics are calculated based on data collected in $\mathtt{E_{124}}$ experiment, **considering the entire area of the experiment, which is 8.08 meters long**.

Comparing Table 7.22 and Table 7.23, we can underscore the importance of the length of the area of interest. These tables contain metrics calculated from the data collected during the $\mathtt{E_{118}}$ experiment, considering the entire and reduced areas of the S8 - ExpEnv4, respectively.

At longer distances from the anchor baseline, the error in distance between people increases significantly. Considering the entire area (17.08 meters long) of the S8 - ExpEnv4 environment, the average **MAE** is 0.23 meters and the **Max** is 0.7 meters. In contrast, considering the reduced area (10.08 meters long) of the S8 - ExpEnv4 environment, the average **MAE** is 0.16 meters and the **Max** is 0.45 meters, which is almost 30% smaller.

This error is not only caused by longer distances from the anchor baseline, but also by the signal interference caused by the electrical enclose, as discussed in Section 3.4.3.

Additionally, considering the entire area (8.08 meters long) of the $\mathtt{E_{118}}$ experiment, the average **MAE** is 0.069 meters and the **Max** is 0.162 meters. This demonstrates that the error significantly decreases together with the length of the area of an environment.

This assertion is further supported by the box plots shown in Figure A.31, Figure A.32 and Figure A.33. These box plots show the distribution of errors in distances between people estimated by Pixel-to-Real method compared to the corresponding distances between people estimated by UWB method. It is important to note that these plots contains negative and positive distance error values.

Comparing the Figure A.31 to the Figure A.32, we can observe that the distance error converge towards zero from the both positive and negative sides. In the full area of $\mathtt{E_{118}}$ (17.08 meters long), the distribution of errors shows more significant deviation, while in the reduced area (10.08 meters long), the errors are closer to zero. This trend is even more visible in the smaller area of the $\mathtt{E_{124}}$ experiment (8.08 meters long), where the distance errors are minimal (see Figure A.33).

This indicates that the accuracy of the model improves as the area of experiment environment is reduced, converging towards zero error in smaller environments. It is also interesting to note that in the S8 - ExpEnv4 the distances between people are more underestimated (distance errors are in most negative), while in the S301 - ExpEnv3 they are more overestimated (distance errors are in most positive). This discrepancy may be due to differences in environmental factors such as light conditions, signal reflection, and interference, which can affect the accuracy of Pixel-to-Real model differently in various environments.

8 Conclusion

In this work, we have proposed a method for collecting automatically annotated data and collected an extensive set of videos with automatically annotated locations of people. This proposal aims to improve the field of people localization and addresses the lack of videos with high-quality annotations. Using the collected data, we have trained a Pixel-to-Real model, which aims to extend the capabilities of static and indoor CCTV cameras to allow for people localization and measuring the distance between them, solely based on the input video data.

We identified weaknesses of the existing localization systems. Many of them require extra hardware, which can be expensive, especially for small organizations. We also noticed a lack of standardization and benchmarking of existing people localization techniques, making it difficult to compare different systems. Throughout the work, we have been striving to resolve these issues.

To collect the annotated video datasets, we developed our own UWB localization system from scratch and improved its precision using the developed by us *Indoor Positioning System* application. During the development of the UWB localization system, we addressed several challenges related to working with IoT devices and gained an invaluable knowledge that will save our time when deploying the entire system for a customer.

We resolved synchronization between the videos and UWB measurements by using the UNIX timestamps since Epoch, which allowed us to precisely align video frames with the closest in time UWB measurements.

By synchronizing the videos with UWB measurements, we acquired a dataset, which we then used to train the Pixel-to-Real model.

We also developed the Optical method, which estimates people's positions using their heights and camera intrinsic parameters. We used this method to evaluate the Pixel-to-Real model.

In the end, we conducted a comprehensive analysis and evaluation of the developed UWB, Pixel-to-Real, and Optical people localization methods by comparing them with reference Ground Truth people positions. By doing so, we demonstrated that the *Indoor Positioning System* application can help to develop and evaluate any people localization method by allowing to align and export the estimations produced by different methods.

We designed our *Indoor Positioning System* application specifically to facilitate the deployment of the Pixel-to-Real model at customer's premises. With the knowledge and expertise gained throughout the work, we expect to deploy our system within two working days.

Through extensive examining of our system in different environments, we proved that our concept ensures repeatability, which is crucial for future improvement in the filed of people localization.

Future work

We plan to increase the the frequency of the position estimations produced by the UWB system. This problem can be solved either by improving the firmware for tags and anchors, or implementing the interpolation technique to fill the gap between the subsequent UWB measurements.

To address the discrepancies in UWB measurements discussed in Section 3.1.1, we attempted to dynamically adjust the antenna delay during the communication stage between a tag and anchors in order to correct the measured distances. The idea was to calculate the antenna delay for each distance interval during the calibration stage and then automatically adjust it during the communication stage. However, this attempt was unsuccessful. We plan to improve this approach in the future, as it will allow to collect more precise distance measurements.

Although, our *Indoor Positioning System* application does not provide direct capability to train the model, we plan to add this feature to make the system more user-friendly.

In the Section 7.4.2, we have discovered that Pixel-to-Real model produces an error in estimations compared to UWB positions. This highlights the need for more accurate UWB positions, which we plan to improve in the future.

Additionally, we plan to integrate people recognition and tracking to ensure correct tag-person association for more precised evaluation of our system.

Finally, we plan to integrate our localization system with other technologies, such as asset tracking, climate control systems, assistance to people with disabilities and more. This will expand the application area of our system.

A Appendix A: Experiment Abbreviations

To structure and name the different experiments performed during our research, we introduce a compact identification system that includes key aspects of each experiment:

Abbreviation format for experiment description

`ExperimentIdentifier(ExperimentType_Environment_Scenario(Tags_ Anchors_TagPlacement_MovementType_WalkLine))`

Where:

- **ExperimentIdentifier**: An identifier of an experiment
  - Format: $\texttt{E}_{\texttt{\{id\}}}$
  - Example: $\texttt{E}_1$.
  - Corresponds to the number of the experiment located in the main repository of this thesis on GitHub[1] .
- **ExperimentType**: A type of an experiment.
  - Possible values: { `Calib`, `DA` }
  - `Calib`: Calibration of the UWB system
  - `DA`: Data acquisition for model training
- **Environment**: An identifier of an environment
  - Possible values: { `Dorm`, `Rot`, `S301`, `S8` }
  - `Dorm`: Dormitory - Experiment Environment 1 (Dorm - ExpEnv1)
  - `Rot`: Rotunda - Experiment Environment 2 (Rot - ExpEnv2)
  - `S301`: S301 - Experiment Environment 3 (S301 - ExpEnv3)
  - `S8`: S8 - Experiment Environment 4 (S8 - ExpEnv4)
- **Scenario**: Scenario ID
  - Format: $\texttt{S}_{\texttt{\{id\}}}$
  - Example: $\texttt{S}_1$
- **Tags**: Number of tags used
  - Format: `T{Number}`
  - Example: `T3`
- **Anchors**: Number of anchors used

[1]https://github.com/Razyapoo/Master-Thesis/tree/main/Recorded Experiments

- Format: `A{Number}`
- Example: `A2`

- **TagPlacement**: The placement of a tag
  - Format: $\texttt{TP}_{\texttt{\{placement\_type\}}}$
  - Possible values: { $\texttt{TP}_\texttt{p}$, $\texttt{TP}_\texttt{h}$ }
  - $\texttt{TP}_\texttt{p}$: Tags placed on pipes
  - $\texttt{TP}_\texttt{h}$: Tags placed on hands
  - **Note:** Initially, anchors were placed on cabinets, but in very first experiments. However, in later experiments, they were always placed on pipes. Therefore, by default, anchors should be considered as placed on pipes unless stated otherwise in the text.

- **MovementType**: A type of a movement during an experiment.
  - Format: $\texttt{M}_{\texttt{\{movement\_type\}}}$
  - Possible values: { $\texttt{M}_\texttt{s}$, $\texttt{M}_\texttt{d}$ }
  - $\texttt{M}_\texttt{s}$: Static experiment
  - $\texttt{M}_\texttt{d}$: Dynamic experiment
  - **Note:** For data acquisition, this is always set as `Dynamic`. Static experiments are performed only during calibration process.

- **WalkLine**: Specifies the walking pattern
  - Format: $\texttt{W}_{\texttt{\{walking\_type\}}}$
  - Possible values: { $\texttt{W}_\texttt{p}$, $\texttt{W}_\texttt{n}$, $\texttt{W}_\texttt{pn}$, $\texttt{W}_\texttt{f}$ }
  - $\texttt{W}_\texttt{p}$: Walking on predefined lines (paper stickers)
  - $\texttt{W}_\texttt{n}$: Walking next to predefined lines
  - $\texttt{W}_\texttt{pn}$: Walking on both predefined lines and next to them
  - $\texttt{W}_\texttt{f}$: Free walking
  - **Note:** For static experiments during the calibration of the UWB system, Walk defines the lines on which the tags are placed. For dynamic experiments, especially during the data acquisition process, Walk defines the walking pattern.

These abbreviations are always described in the text in the place where they appear. These abbreviations are intended to facilitate orientation in performed experiments. An example of the abbreviation is as follows:

- $\texttt{E}_{\texttt{83}}\texttt{(Calib\_Dorm\_S}_\texttt{3}\texttt{(T3\_A2\_TP}_\texttt{p}\texttt{\_M}_\texttt{d}\texttt{\_W}_\texttt{p}\texttt{))}$: This experiment represents a calibration `Calib` conducted in the Dorm - ExpEnv1 environment; it follows the $\texttt{S}_\texttt{3}$ scenario and involves the use of 1 tag and 2 anchors; tags are placed on pipes ($\texttt{TP}_\texttt{p}$); the experiment is static ($\texttt{M}_\texttt{s}$) with pipes are placed on predefined lines only ($\texttt{W}_\texttt{p}$).

- $\mathtt{E}_{115}$`(DA_S8_`$\mathtt{S}_5$`(T1_A2_`$\mathtt{TP}_\mathtt{h}$`_`$\mathtt{M}_\mathtt{d}$`_`$\mathtt{W}_\mathtt{pn}$`)):` This experiment represents a data acquisition `DA` conducted in the S8 - ExpEnv4 environment; it follows the $\mathtt{S}_5$ scenario and involves the use of 1 tag and 2 anchors; tags are placed on pipes ($\mathtt{TP}_\mathtt{p}$); the experiment is static ($\mathtt{M}_\mathtt{s}$) with pipes are placed on predefined lines and next to them ($\mathtt{W}_\mathtt{pn}$).

Abbreviation format for datasets

Video Data

- `VD(expid, [start_time, end_time])`
  `→ { (frameid, frametimestamp) }`
  - Function returning a video data for a given experiment ID and an optional specific range of timestamps.
  - Tag ID uniquely identifies a person.
  - The result is a tuple, consisting of video frame ID and timestamp in milliseconds since the Unix Epoch.
  - If `start_time` and `end_time` are not specified, the function returns data for the entire range of timestamps (video duration).

Distances

- $\mathtt{D}_{\mathtt{\{type\}}}$`(expid, [tagid], [start_time, end_time])`
  `→ { timestamp, tagid, anchorid1, distance1, anchorid2, distance2 }`
  - Function returning a set of **raw**, or **corrected** distances collected using UWB technology for a given experiment ID, an optional Tag ID, and an optional specific range of timestamps.
  - `type` can be either `raw` for uncorrected data or `corr` for corrected data.
  - Tag ID uniquely identifies a person.
  - This represents data measurements before (`raw`) or after (`corr`) correction.
  - The returned values consist of timestamp of the measurement, tag identifier, anchor identifier, and distances from each anchor.
  - If `tagid` is not specified, the function returns coordinates of all tags.
  - If `start_time` and `end_time` are not specified, the function returns coordinates for the entire range of video.
- $\mathtt{D}_{\mathtt{\{type\}}}$`(expid, tagid, anchorid, [start_time, end_time])`
  `→ { distance }`

- Function returning a set of **raw** or **corrected** distances for a given experiment ID, Tag ID and Anchor ID pair, and an optional specific range of timestamps.
- `type` can be either `raw` for uncorrected data or `corr` for corrected data.
- Tag ID uniquely identifies a person.
- This represents data measurements before (`raw`) or after (`corr`) correction.
- If `tagid` is not specified, the function returns coordinates of all tags.
- If `start_time` and `end_time` are not specified, the function returns coordinates for the entire range of video.

- $\texttt{DPeople}_{\texttt{\{method\}}}\texttt{(tagid\_1, tagid\_2)} \to \texttt{\{ distance}_{\texttt{\{frameid\}}}\texttt{ \}}$

  - Function calculating the Euclidean distance between two sets of coordinates $\texttt{C}_{\texttt{\{method\}}}\texttt{(expid, tagid\_1)}$ and $\texttt{C}_{\texttt{\{method\}}}\texttt{(expid, tagid\_2)}$, where `method` can be `UWB`, `P2R`, `Opt`, or `RS`.
  - `method` and `expid` are assumed to be the same. Therefore they are omitted in the main formula. This is a necessary step, because otherwise the formula will not fit into tables.
  - Sets $\texttt{C}_{\texttt{\{method\}}}$ are synchronized by frames. We consider pairs of tags (people) only from the same frame.
  - Returns a set of distances. Each element of the set corresponds to a distance between people in a certain frame.

Coordinates

- $\texttt{C}_{\texttt{\{method\}}}\texttt{(expid, [tagid], [start\_time, end\_time])}$
  $\to \texttt{\{ (x}_{\texttt{\{method\}}}\texttt{, y}_{\texttt{\{method\}}}\texttt{) \}}$

  - Function returning a set of coordinates derived from the specified method (`UWB`, `P2R`, `Opt`) for a given experiment ID, an optional Tag ID, and an optional specific range of timestamps.
  - For `UWB`, the function returns a set of coordinates derived from the **corrected** UWB distances $\texttt{D}_{\texttt{corr}}\texttt{(expid, [tagid], [start\_time, end\_time])}$ for a given experiment ID, an optional Tag ID, and an optional specific range of timestamps.
  - If `tagid` is not specified, the function returns coordinates of all tags.
  - If `start_time` and `end_time` are not specified, the function returns coordinates for the entire range of video.
  - Tag ID uniquely identifies a person.

Reference system

- $\texttt{RS}_{\texttt{\{spacing\}}}\texttt{([tagid])} \to \texttt{\{ (x}_{\texttt{ref}}\texttt{, y}_{\texttt{ref}}\texttt{) \} or \{ d}_{\texttt{ref}}\texttt{ \}}$

- Function returning the reference system for the specified spacing (`1m`, `0.5m`) and an optional Tag ID.
- Tag ID uniquely identifies a person.
- Return values consist of either reference coordinates (x_{ref}, y_{ref}) or reference distances (d_{ref}), spaced according to the specified spacing (`1m`, `0.5m`).
- If `tagid` is specified, the function returns reference points specific to that tag; otherwise, it returns all reference points in the current reference system.

To show that in the experiment $\mathtt{E_{83}(Calib\_Dorm\_S_3(T3\_A2\_TP_p\_M_d\_W_p))}$ we get $\mathtt{D_{raw}(E_{83})}$ dataset, we use the following representation:

$$\mathtt{E_{83}(Calib\_Dorm\_S_3(T3\_A2\_TP_p\_M_d\_W_p))} \rightarrow \mathtt{D_{raw}(E_{83})}$$

A Appendix B: Set of images

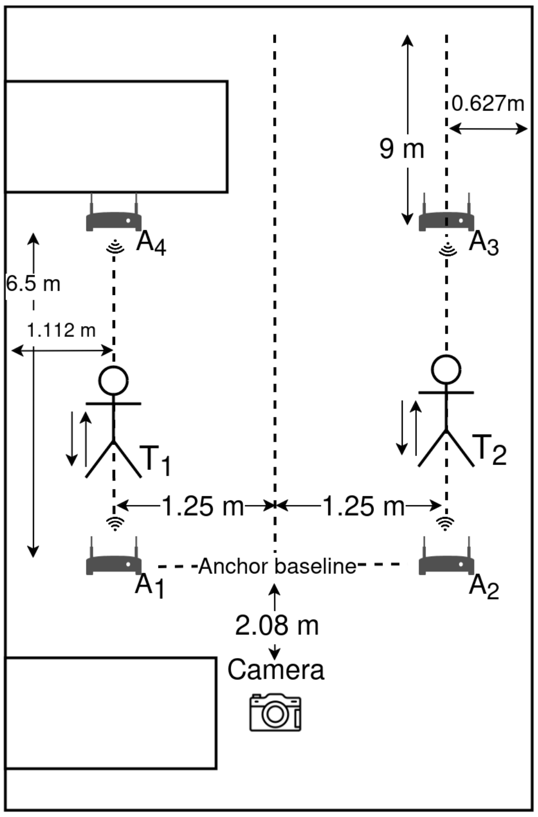


Figure A.1 Schema of the S301 - Experiment Environment 3

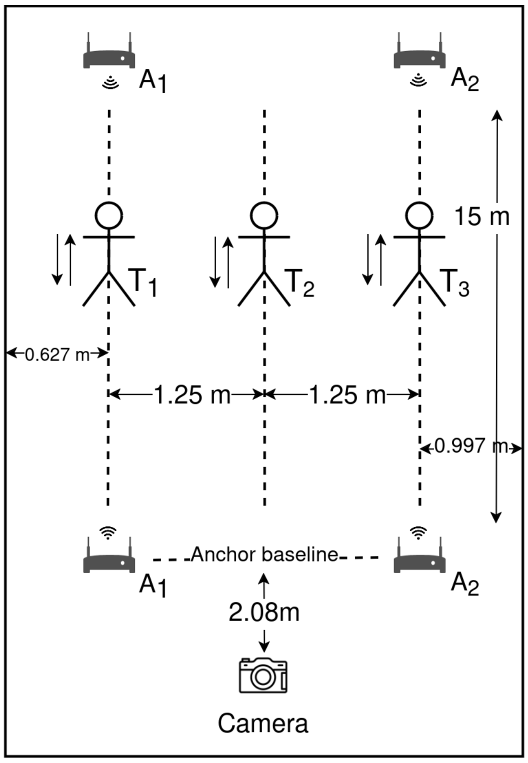


Figure A.2 Schema of the S8 - Experiment Environment 4

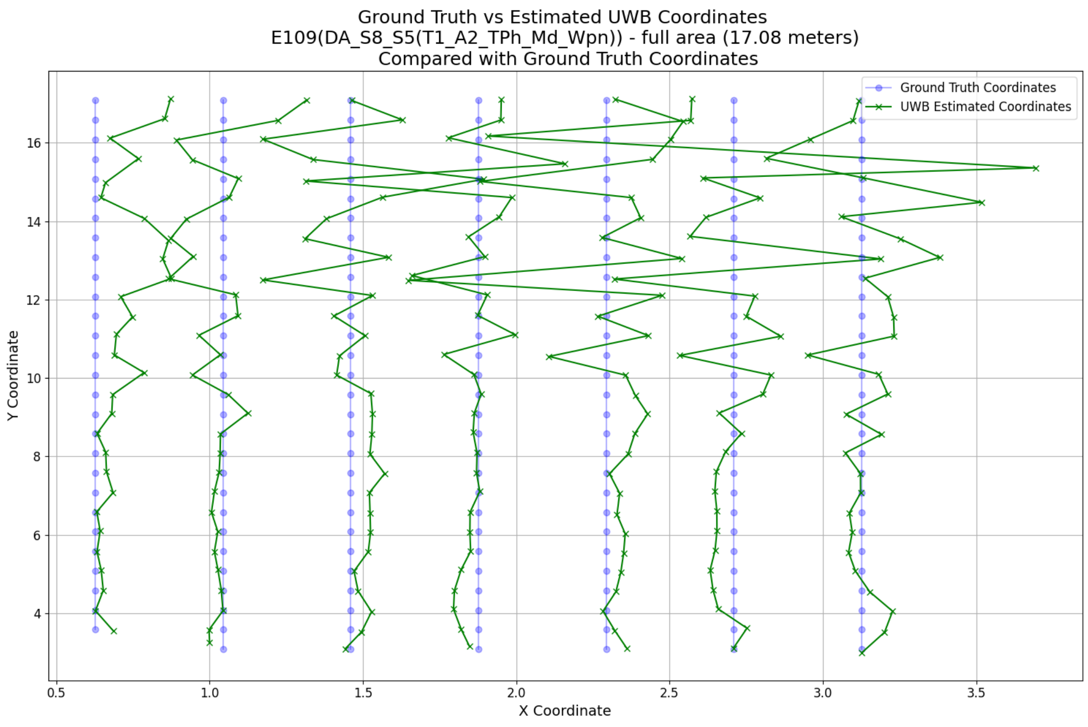


Figure A.3 The scatter plot illustrates the correlation between the estimated UWB coordinates (green) and the reference Ground Truth coordinates (blue). This data is collected in $\mathtt{E}_{109}$, considering the entire area of the environment, which is 17.08 meters long.

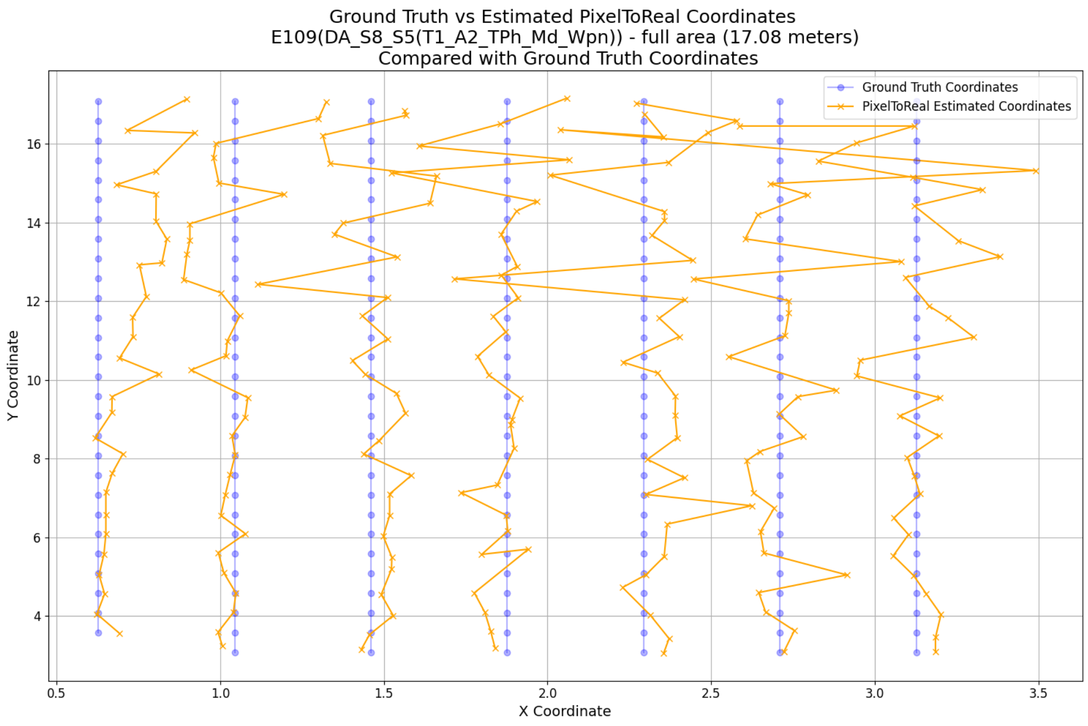


Figure A.4 The scatter plot illustrates the correlation between coordinates predicted by Pixel-to-Real method (yellow) and the reference Ground Truth coordinates (blue). This data is collected in $\mathtt{E}_{109}$, considering the entire area of the environment, which is 17.08 meters long.

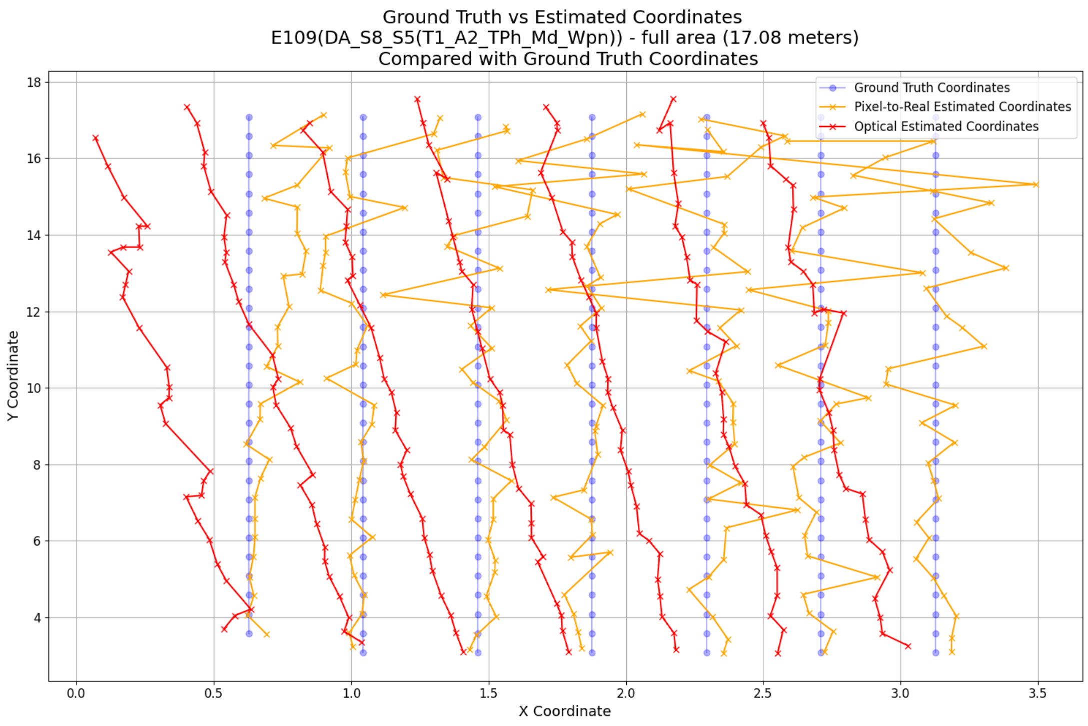


Figure A.5 The scatter plot illustrates the comparison of coordinates estimated using Pixel-to-Real model (yellow) and coordinates estimated using Optical method (red) with respect to the reference Ground Truth coordinates (blue). This data is collected in $\mathtt{E}_{109}$.

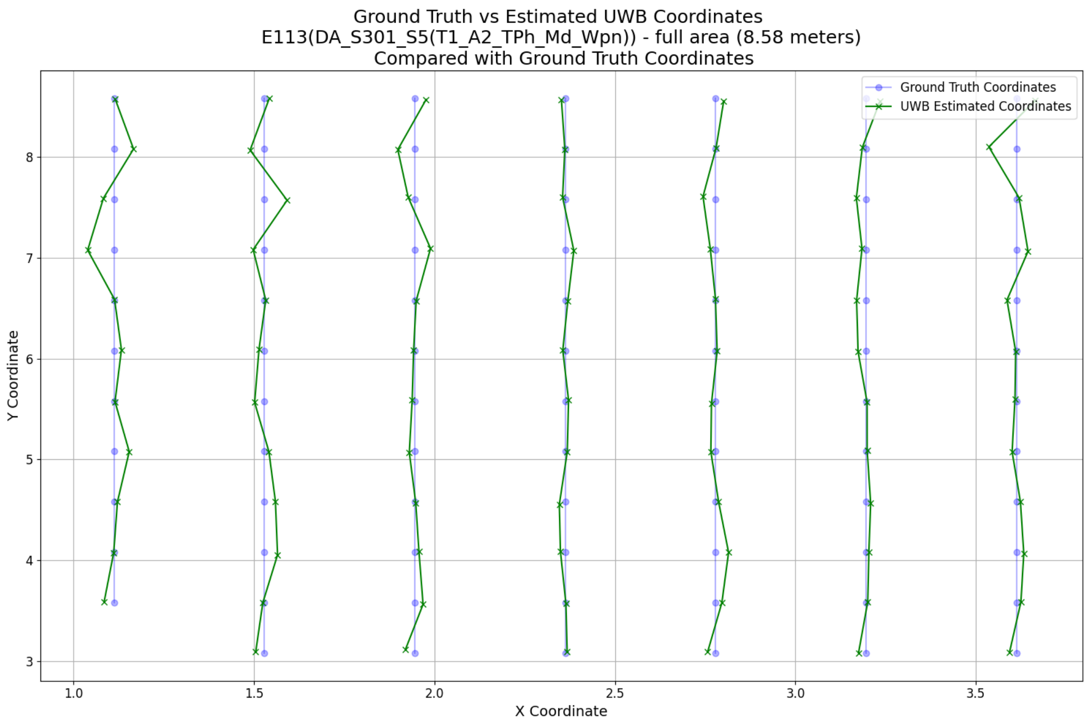


Figure A.6 The scatter plot illustrates the correlation between the estimated UWB coordinates (green) and the reference Ground Truth coordinates (blue). This data is collected in $\texttt{E}_{113}$, considering the entire area of the environment, which is 8.58 meters long.

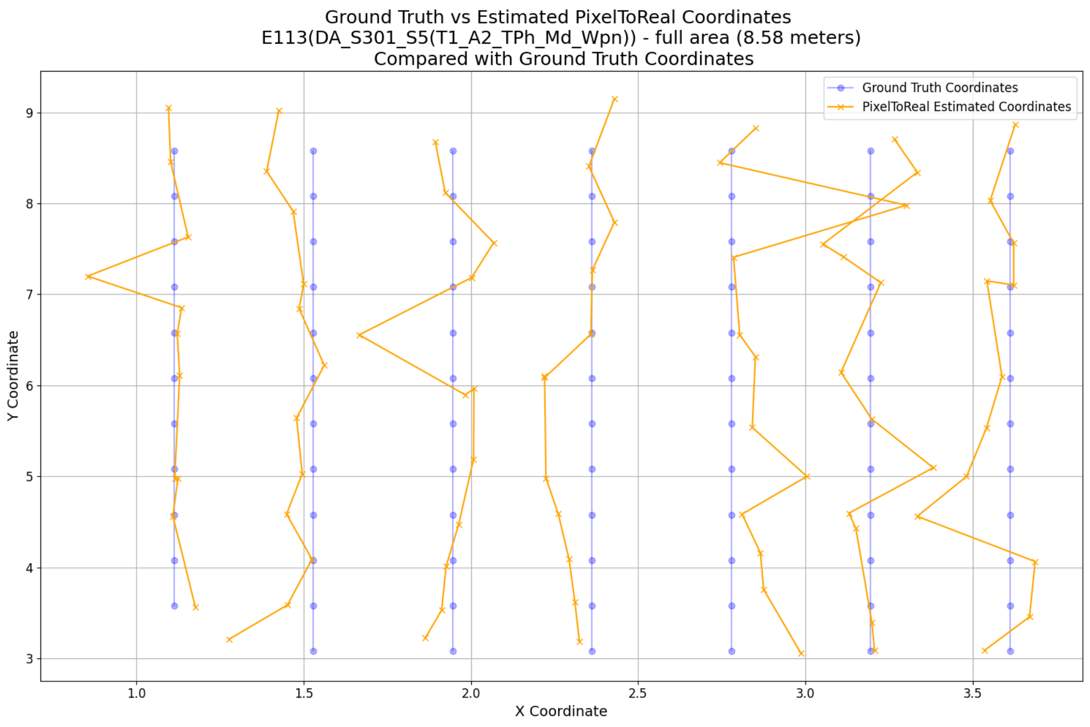


Figure A.7 The scatter plot illustrates the correlation between coordinates predicted by Pixel-to-Real method (yellow) and the reference Ground Truth coordinates (blue). This data is collected in $\texttt{E}_{113}$, considering the entire area of the environment, which is 8.58 meters long.

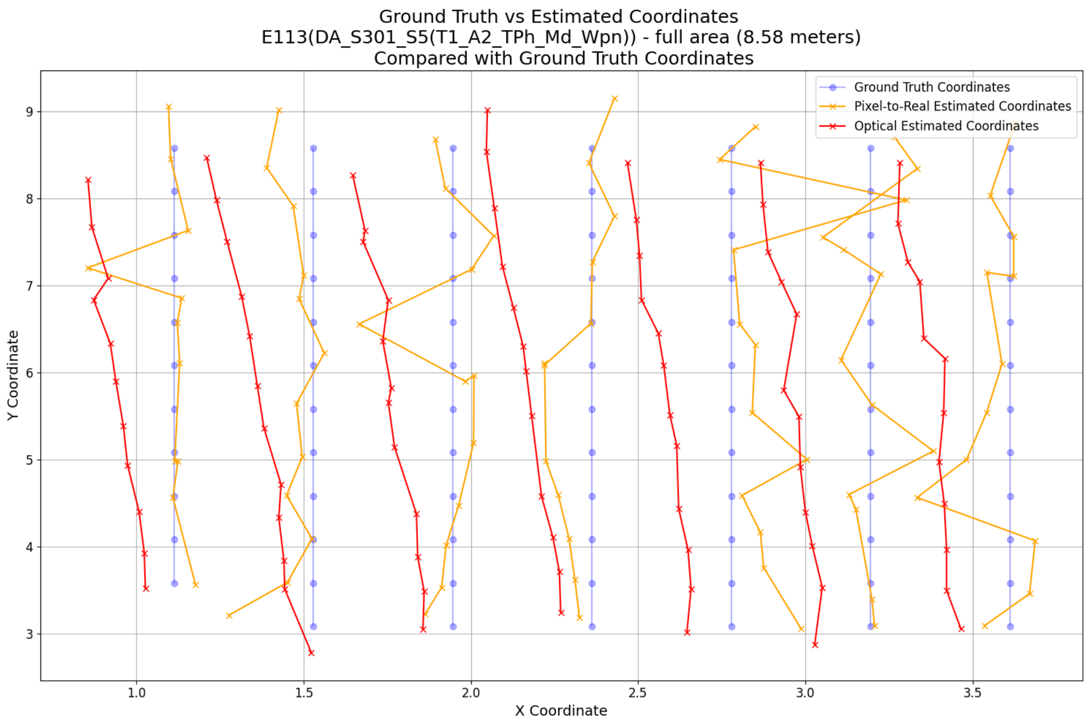


Figure A.8 The scatter plot illustrates the comparison of coordinates estimated using Pixel-to-Real model (yellow) and coordinates estimated using Optical method (red) with respect to the reference Ground Truth coordinates (blue). This data is collected in $\texttt{E}_{113}$, considering the entire area of the environment, which is 8.58 meters long.

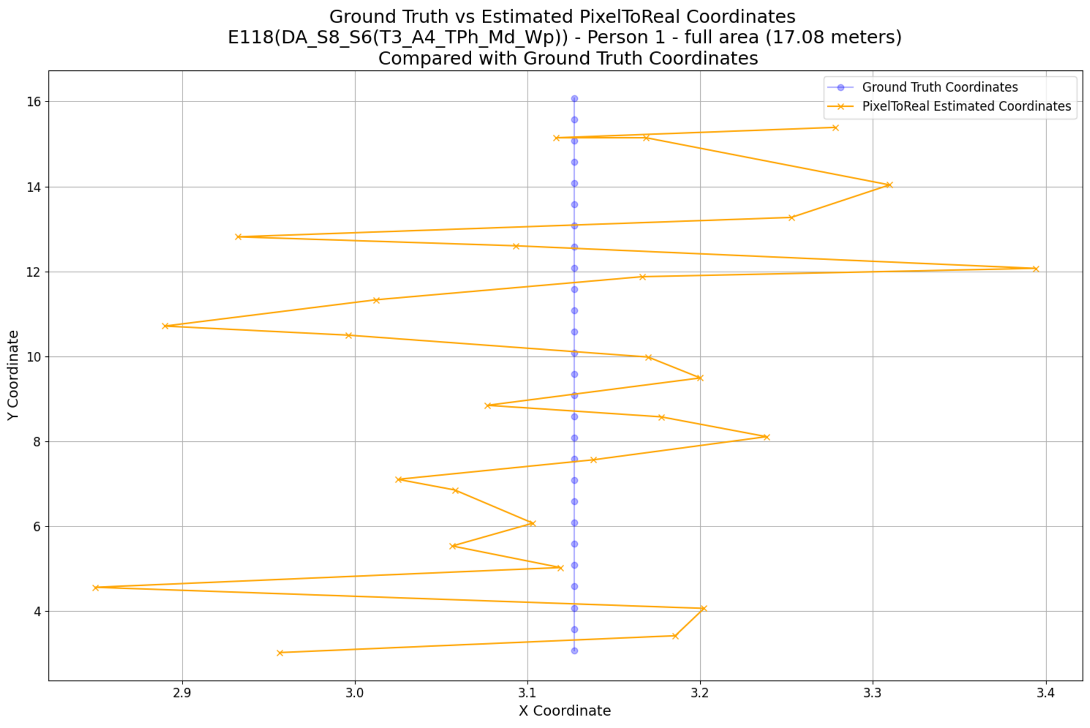


Figure A.9 The scatter plot illustrates the correlation between coordinates predicted by Pixel-to-Real method (yellow) and the reference Ground Truth coordinates (blue). This data is collected in $\texttt{E}_{118}$ for Person 1, considering the entire area of the environment, which is 17.08 meters long. Data and plots for other participants of the $\texttt{E}_{118}$ experiment are available on GitHub.

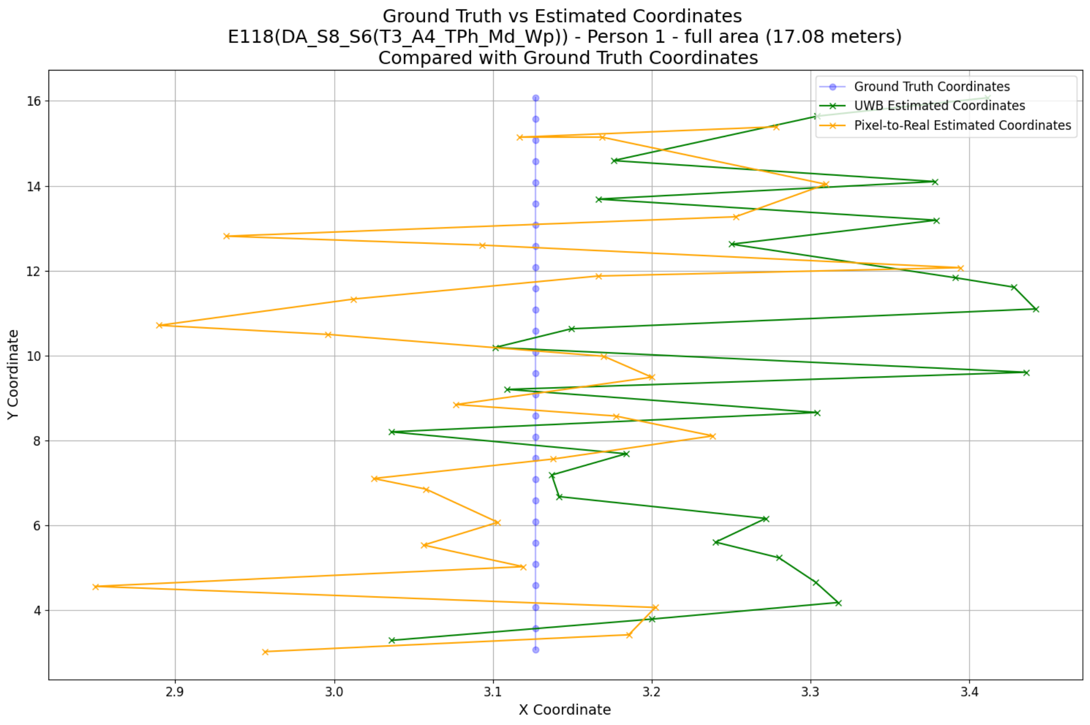


Figure A.10 The scatter plot illustrates the comparison of coordinates estimated using UWB method (green) and coordinates estimated using Pixel-to-Real model (yellow) with respect to the reference Ground Truth coordinates (blue). This data is collected in $\texttt{E}_{118}$ for Person 1, considering the entire area of the environment, which is 17.08 meters long. Data and plots for other participants of the $\texttt{E}_{118}$ experiment are available on GitHub.

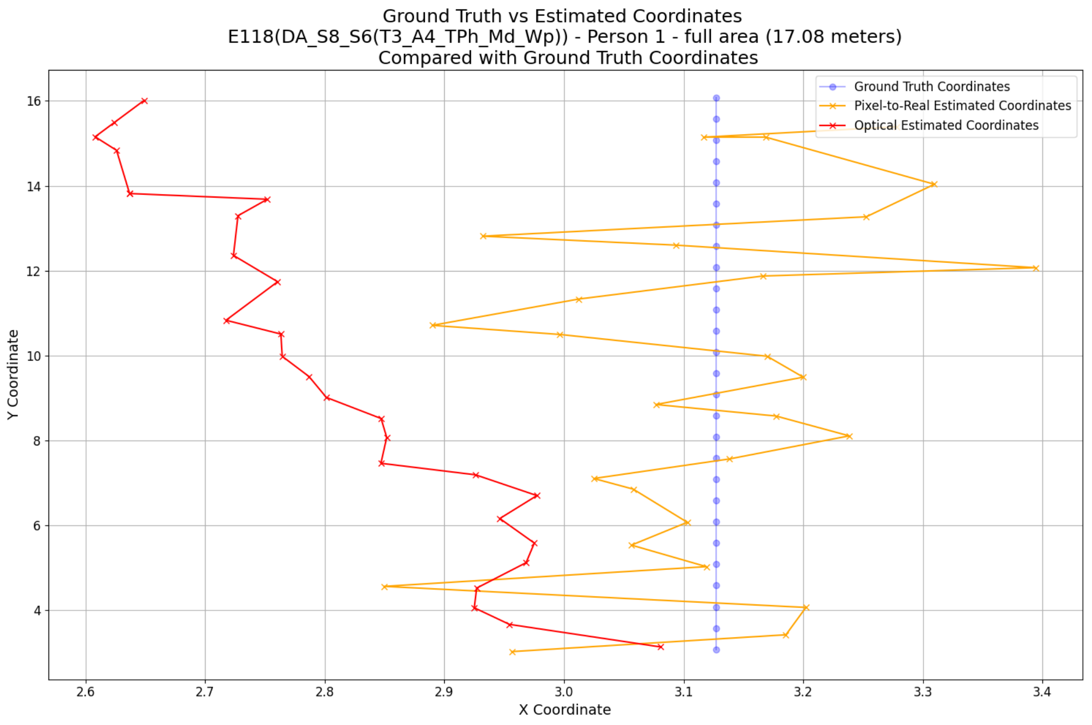


Figure A.11 The scatter plot illustrates the comparison of coordinates estimated using Pixel-to-Real model (yellow) and coordinates estimated using Optical method (red) with respect to the reference Ground Truth coordinates (blue). This data is collected in $\mathtt{E}_{118}$ for Person 1, considering the entire area of the environment, which is 17.08 meters long. Data and plots for other participants of the $\mathtt{E}_{118}$ experiment are available on GitHub.

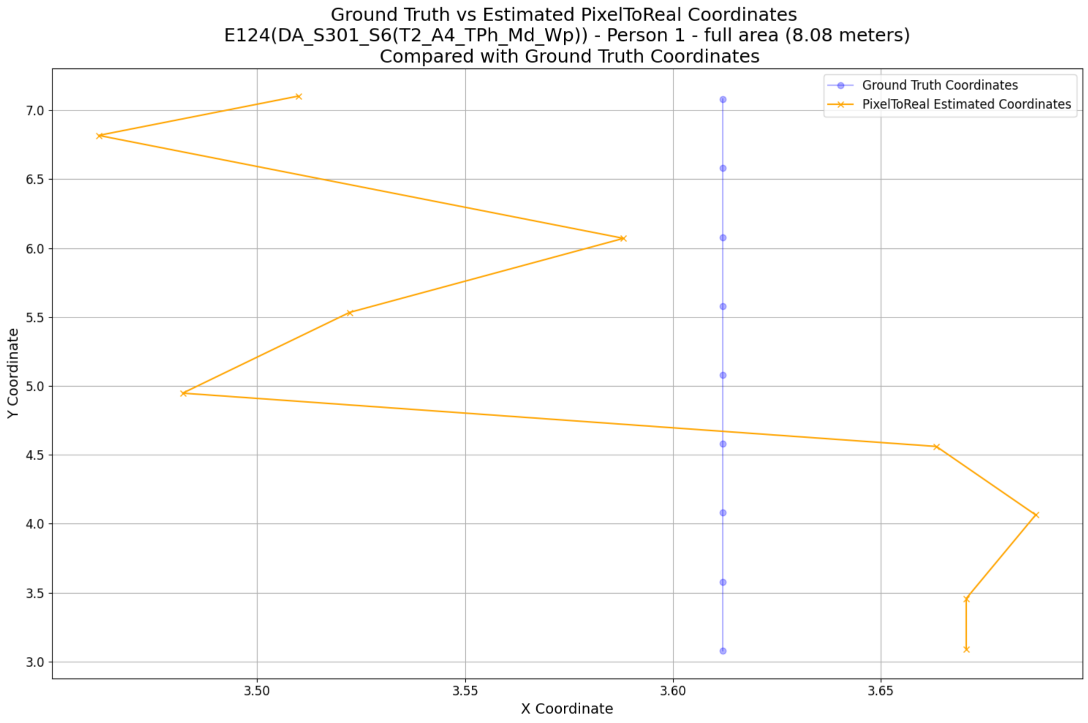


Figure A.12 The scatter plot illustrates the correlation between coordinates predicted by Pixel-to-Real method (yellow) and the reference Ground Truth coordinates (blue). This data is collected in $\mathtt{E}_{124}$ for Person 1, considering the entire area of the environment, which is 8.08 meters long. Data and plots for other participants of the $\mathtt{E}_{124}$ experiment are available on GitHub.

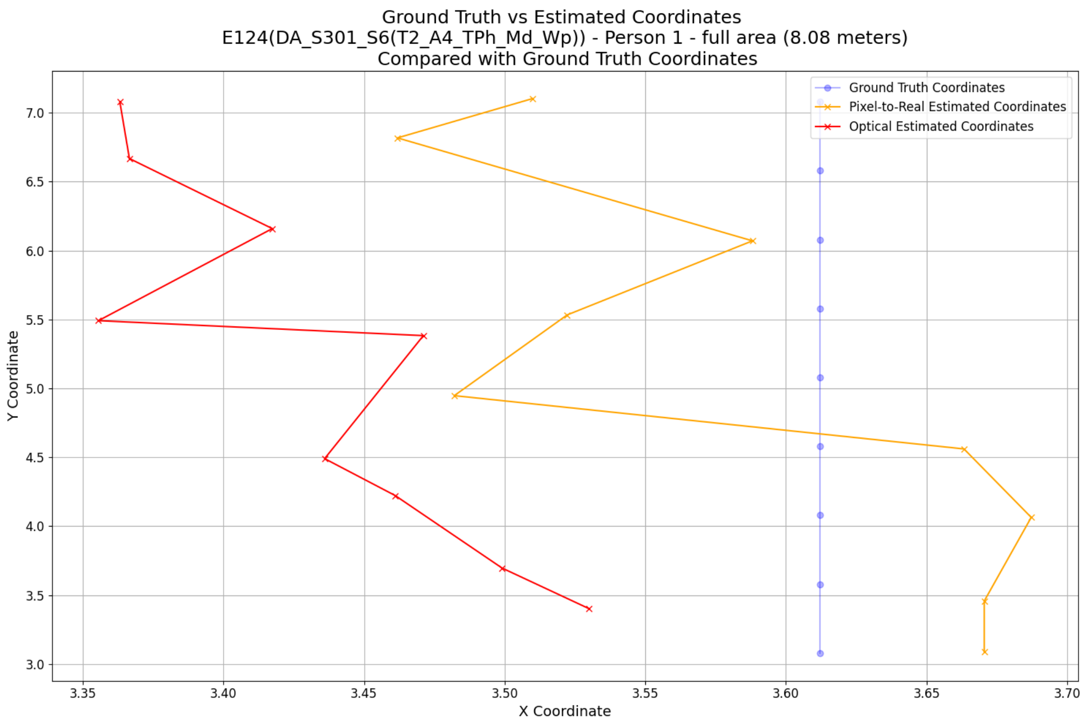


Figure A.13 The scatter plot illustrates the comparison of coordinates estimated using Pixel-to-Real model (yellow) and coordinates estimated using Optical method (red) with respect to the reference Ground Truth coordinates (blue). This data is collected in $\mathtt{E}_{124}$ for Person 1, considering the entire area of the environment, which is 8.08 meters long. Data and plots for other participants of the $\mathtt{E}_{124}$ experiment are available on GitHub.

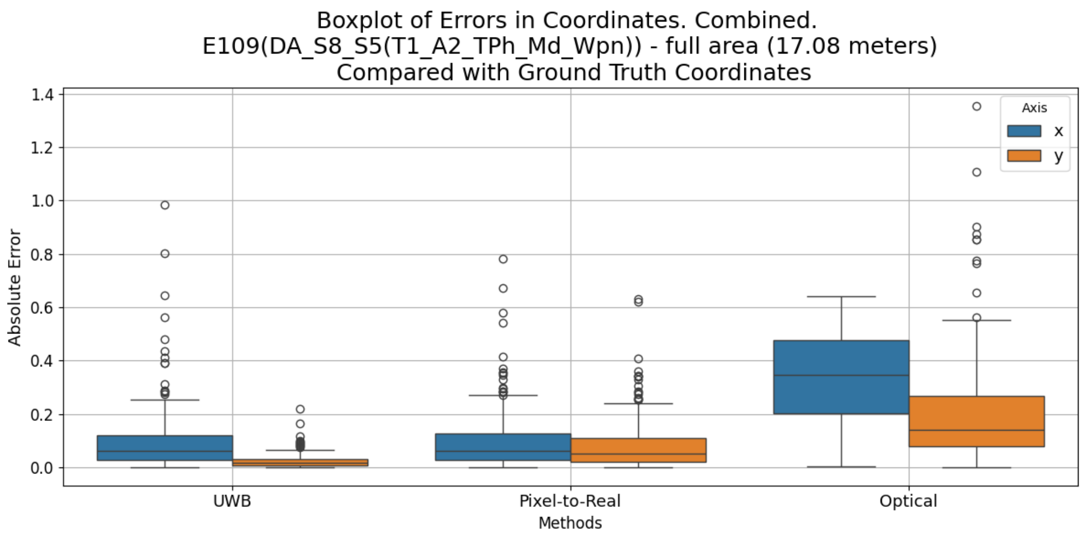


Figure A.14 The boxplot illustrates the distribution of absolute errors in UWB coordinates, as well as coordinates estimated using Pixel-to-Real and Optical methods. This data is collected in $\mathtt{E}_{109}$, considering the entire area of the environment, which is 17.08 meters long.

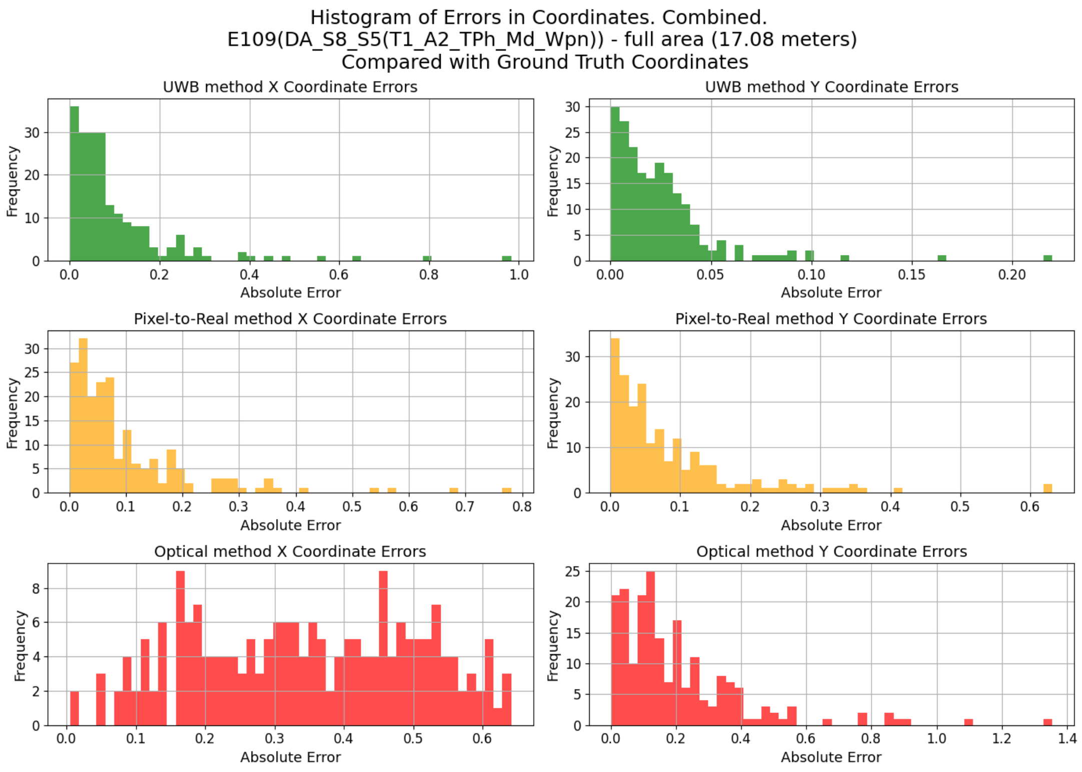


Figure A.15 The histogram plot illustrates the distribution of absolute errors in UWB coordinates, as well as coordinates estimated using Pixel-to-Real and Optical methods. This data is collected in $\mathtt{E}_{109}$, considering the entire area of the environment, which is 17.08 meters long.

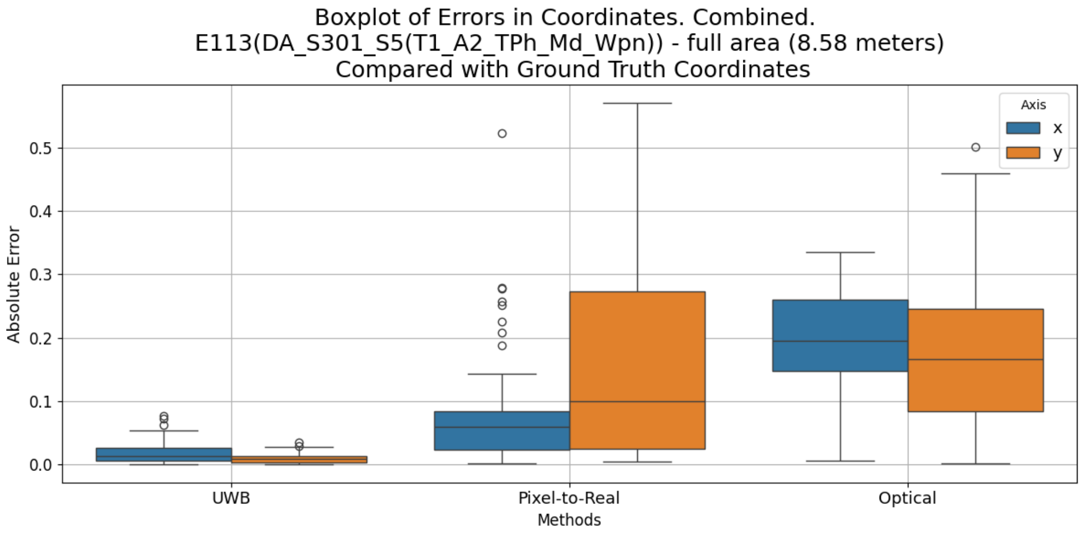


Figure A.16 The boxplot illustrates the distribution of absolute errors in UWB coordinates, as well as coordinates estimated using Pixel-to-Real and Optical methods. This data is collected in $\mathtt{E}_{113}$, considering the entire area of the environment, which is 8.58 meters long.

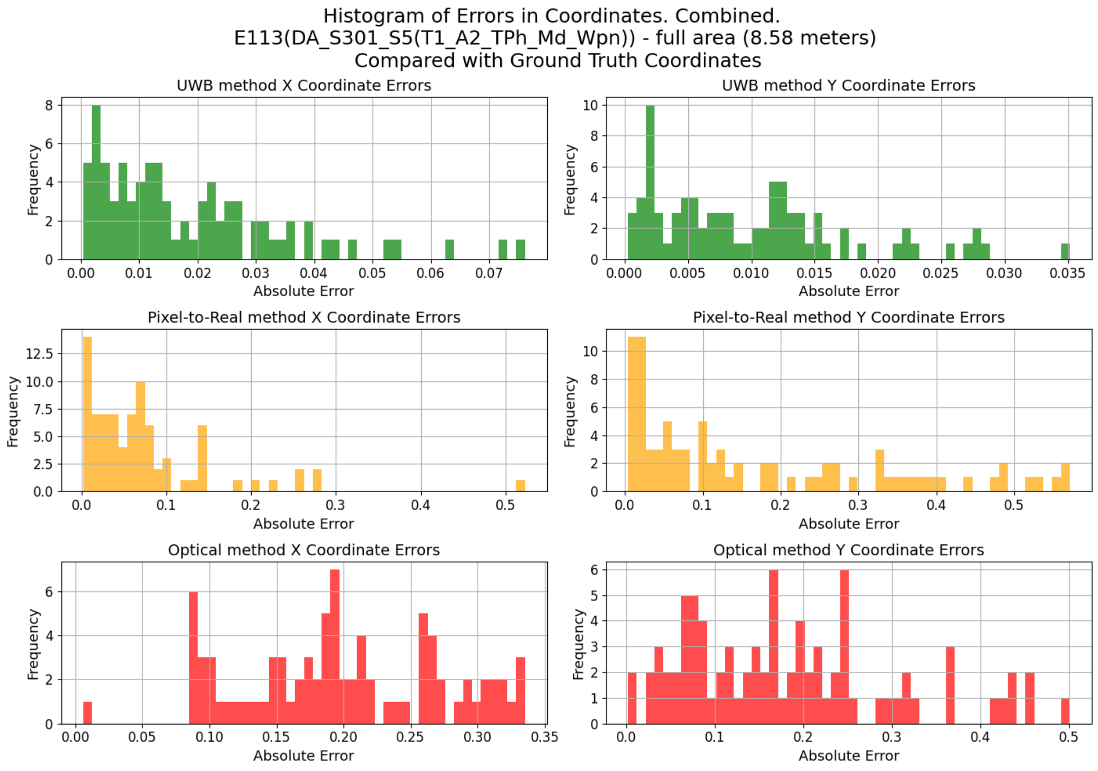


Figure A.17 The histogram plot illustrates the distribution of absolute errors in UWB coordinates, as well as coordinates estimated using Pixel-to-Real and Optical methods. This data is collected in $\mathtt{E}_{113}$, considering the entire area of the environment, which is 8.58 meters long.

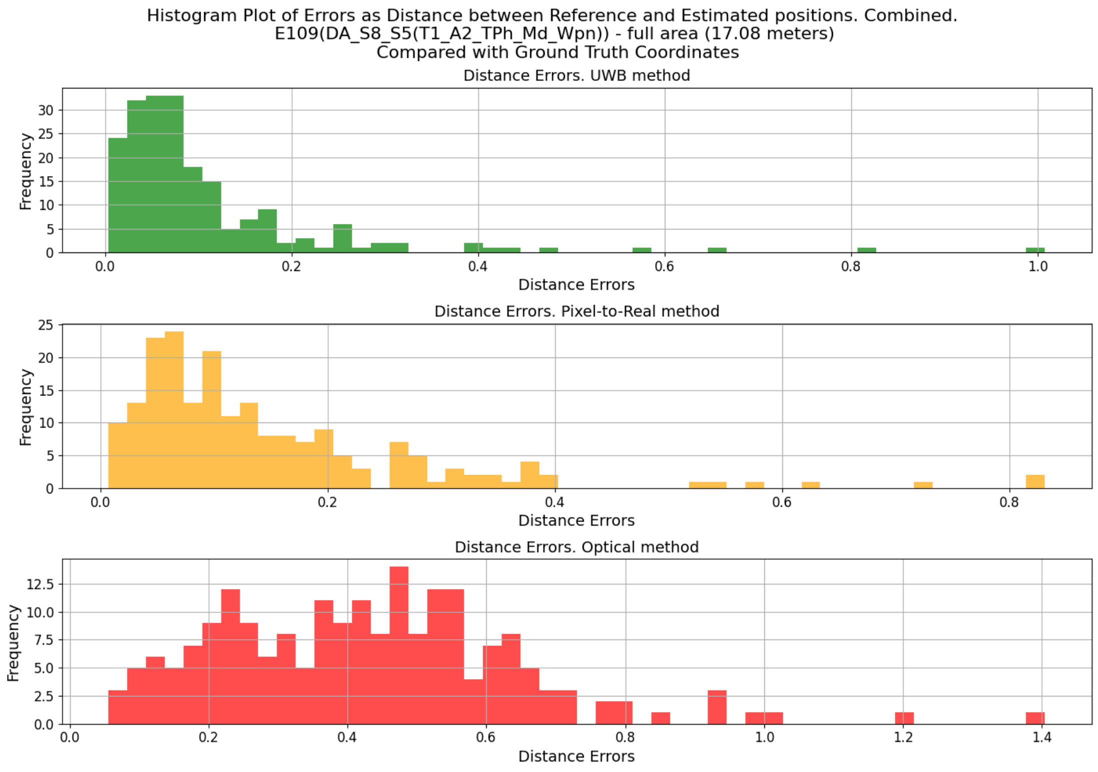


Figure A.18 The histogram plot illustrates the distribution of absolute errors in UWB coordinates (positions), as well as coordinates (positions) estimated using Pixel-to-Real and Optical methods, as Euclidean distances between the estimated coordinates (positions) and the corresponding reference Ground Truth coordinates (positions). This data is collected in $\mathtt{E}_{109}$, considering the entire area of the environment, which is 17.08 meters long.

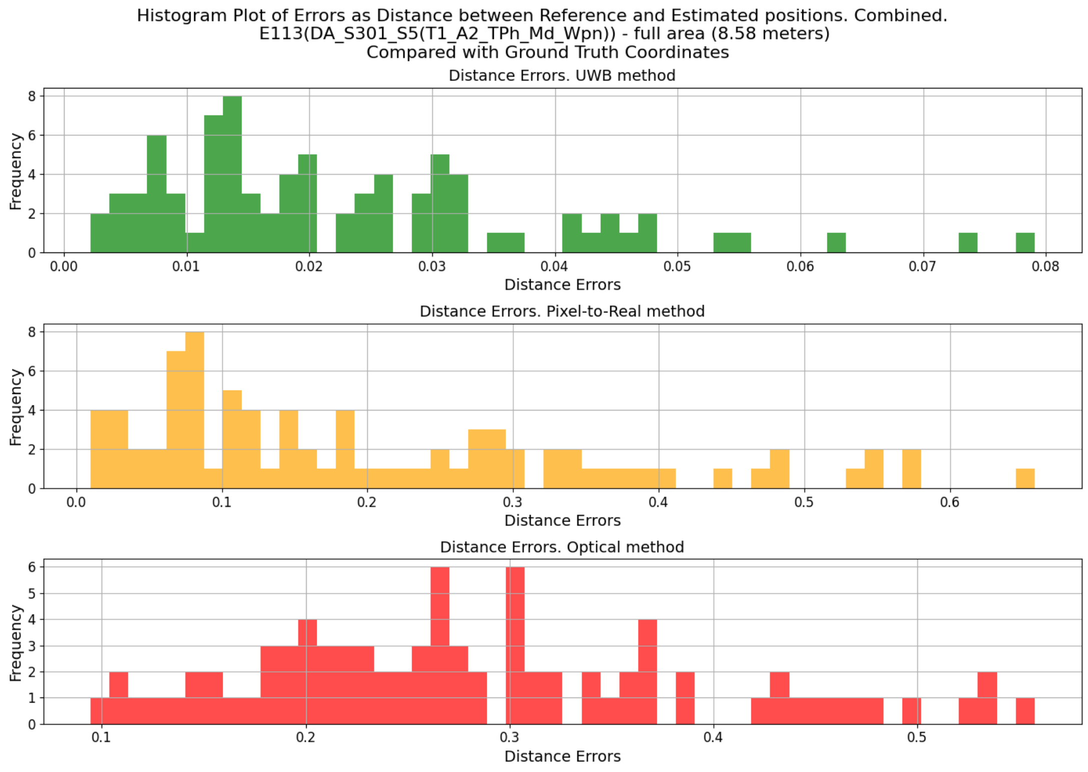


Figure A.19 The histogram plot illustrates the distribution of absolute errors in UWB coordinates (positions), as well as coordinates (positions) estimated using Pixel-to-Real and Optical methods, as Euclidean distances between the estimated coordinates (positions) and the corresponding reference Ground Truth coordinates (positions). This data is collected in $\texttt{E}_{113}$, considering the entire area of the environment, which is 8.58 meters long.

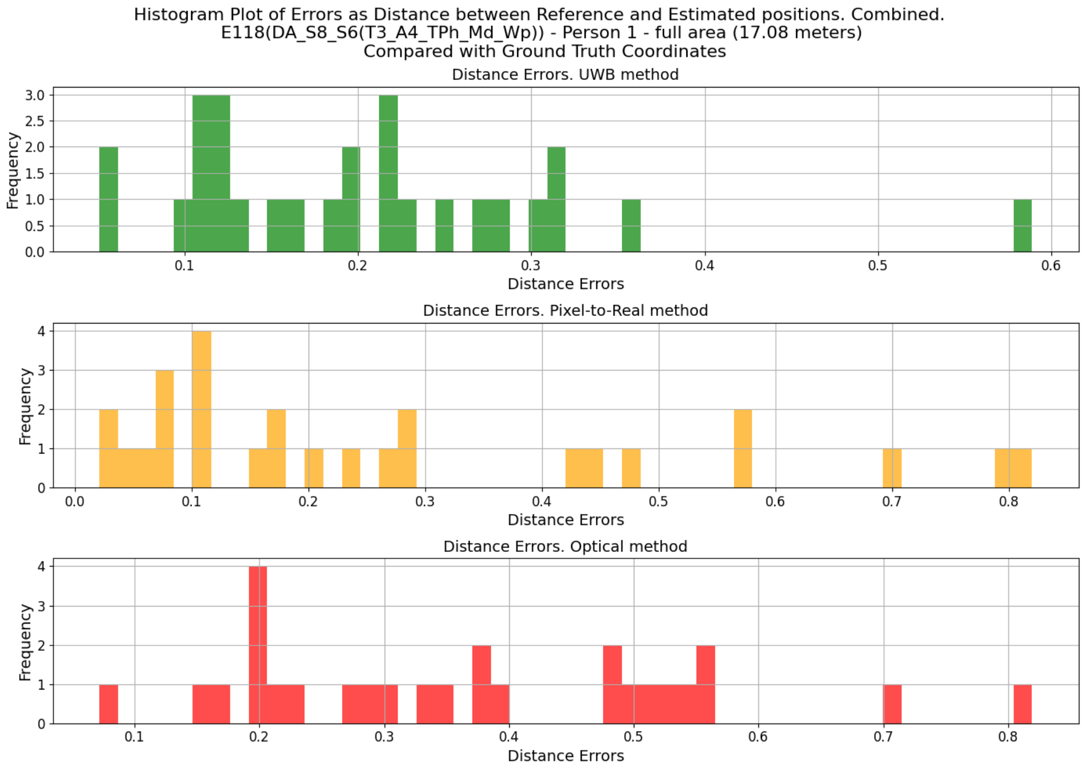


Figure A.20 The histogram plot illustrates the distribution of absolute errors in UWB coordinates (positions), as well as coordinates (positions) estimated using Pixel-to-Real and Optical methods, as Euclidean distances between the estimated coordinates (positions) and the corresponding reference Ground Truth coordinates (positions). This data is collected in $\mathtt{E}_{118}$ for Person 1, considering the entire area of the environment, which is 17.08 meters long. Data and plots for other participants of the $\mathtt{E}_{118}$ experiment are available on GitHub.

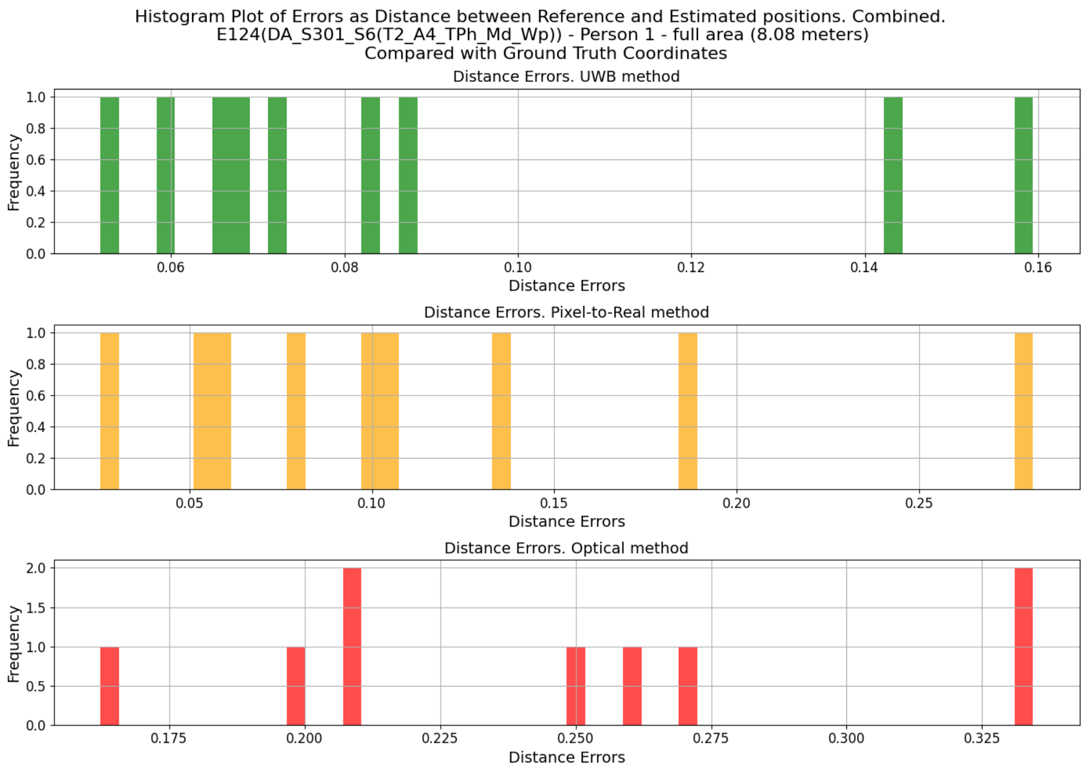


Figure A.21 The histogram plot illustrates the distribution of absolute errors in UWB coordinates (positions), as well as coordinates (positions) estimated using Pixel-to-Real and Optical methods, as Euclidean distances between the estimated coordinates (positions) and the corresponding reference Ground Truth coordinates (positions). This data is collected in $\mathtt{E}_{124}$ for Person 1, considering the entire area of the environment, which is 8.08 meters long. Data and plots for other participants of the $\mathtt{E}_{124}$ experiment are available on GitHub.

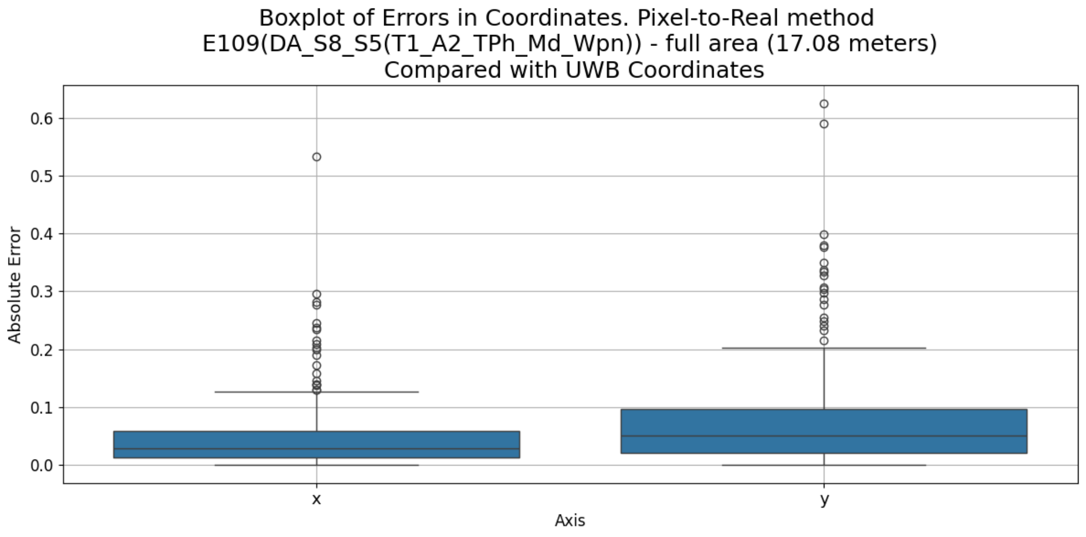


Figure A.22 The boxplot illustrates the distribution of absolute errors in coordinates estimated using Pixel-to-Real method compared to UWB coordinates. This data is collected in $\mathtt{E}_{109}$, considering the entire area of the environment, which is 17.08 meters long.

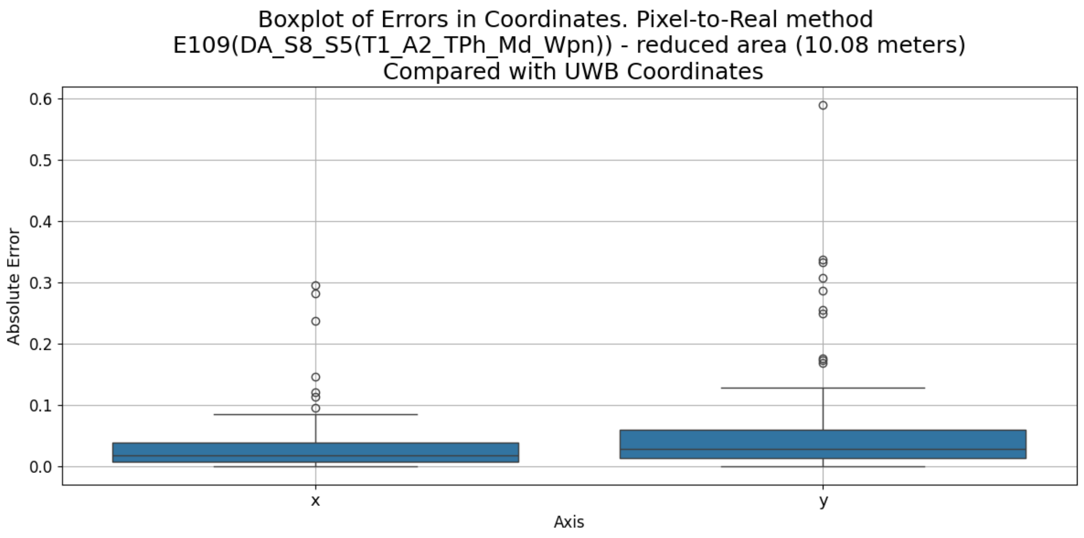


Figure A.23 The boxplot illustrates the distribution of absolute errors in coordinates estimated using Pixel-to-Real method compared to UWB coordinates. This data is collected in $\mathtt{E}_{109}$, considering the reduced area of the environment, which is 10.08 meters long.

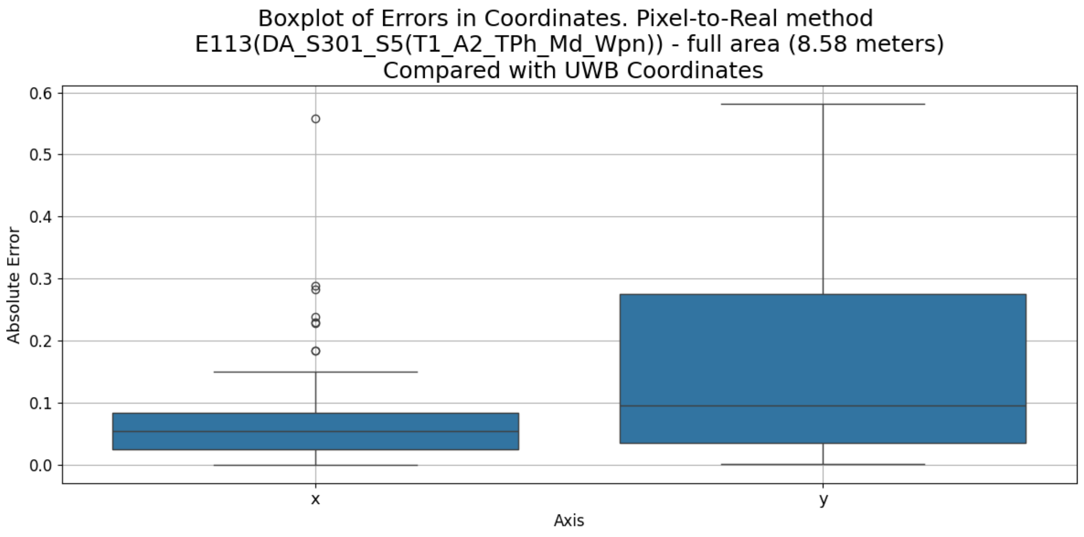


Figure A.24 The boxplot illustrates the distribution of absolute errors in coordinates estimated using Pixel-to-Real method compared to UWB coordinates. This data is collected in $\mathtt{E}_{113}$, considering the entire area of the environment, which is 8.58 meters long.

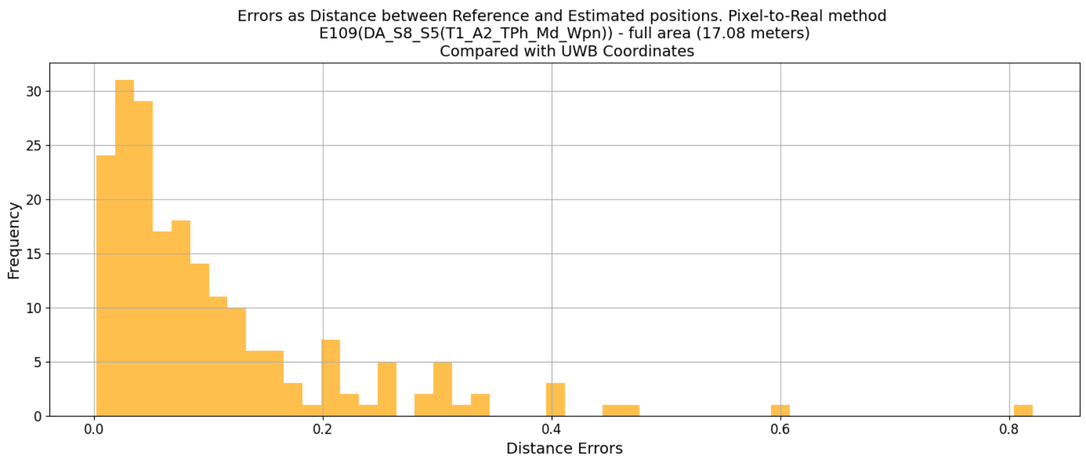


Figure A.25 The histogram illustrates the distribution of absolute errors as a distance between the positions (coordinates) estimated by Pixel-to-Real method and the positions (coordinates) estimated by UWB coordinates. This data is collected in $\mathtt{E}_{109}$, considering the entire area of the environment, which is 17.08 meters long.

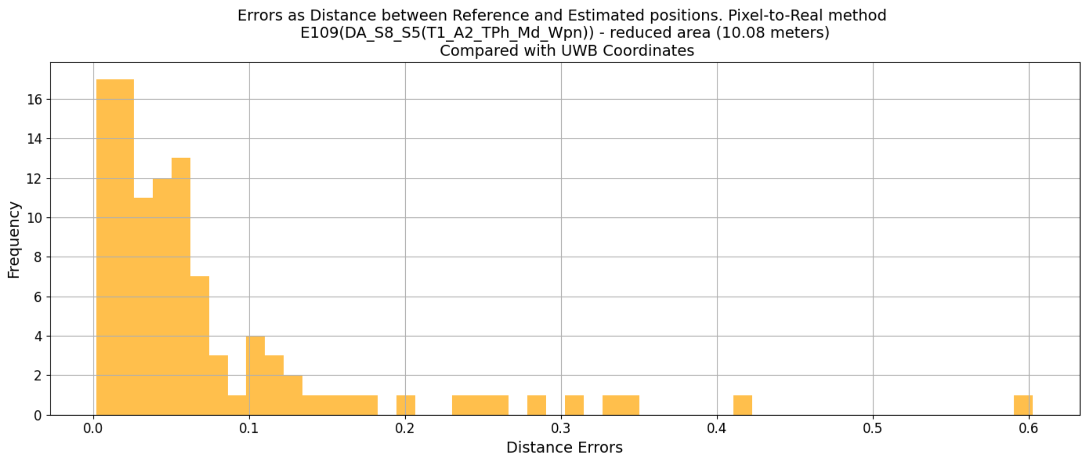


Figure A.26 The histogram illustrates the distribution of absolute errors as a distance between the positions (coordinates) estimated by Pixel-to-Real method and the positions (coordinates) estimated by UWB coordinates. This data is collected in $\mathtt{E}_{109}$, considering the reduced area of the environment, which is 10.08 meters long.

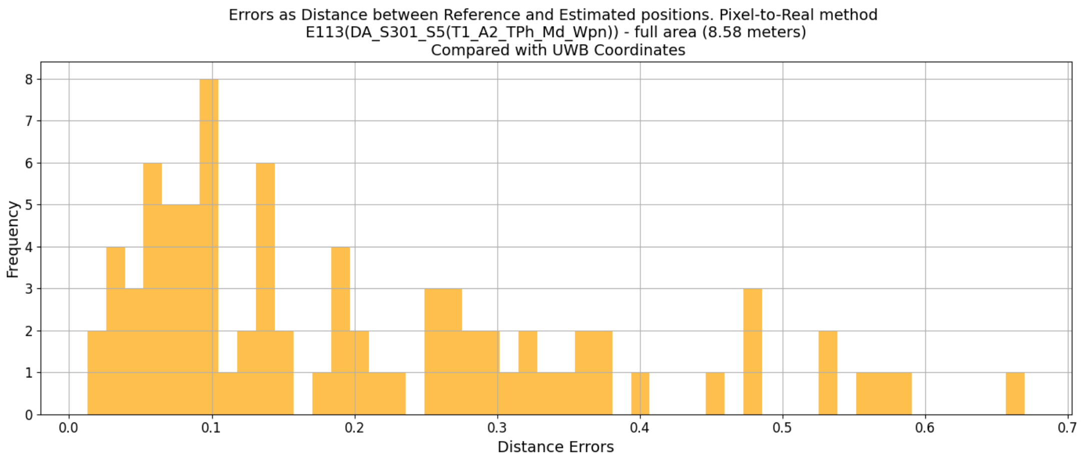


Figure A.27 The histogram illustrates the distribution of absolute errors as a distance between the positions (coordinates) estimated by Pixel-to-Real method and the positions (coordinates) estimated by UWB coordinates. This data is collected in $\texttt{E}_{113}$, considering the entire area of the environment, which is 8.58 meters long.

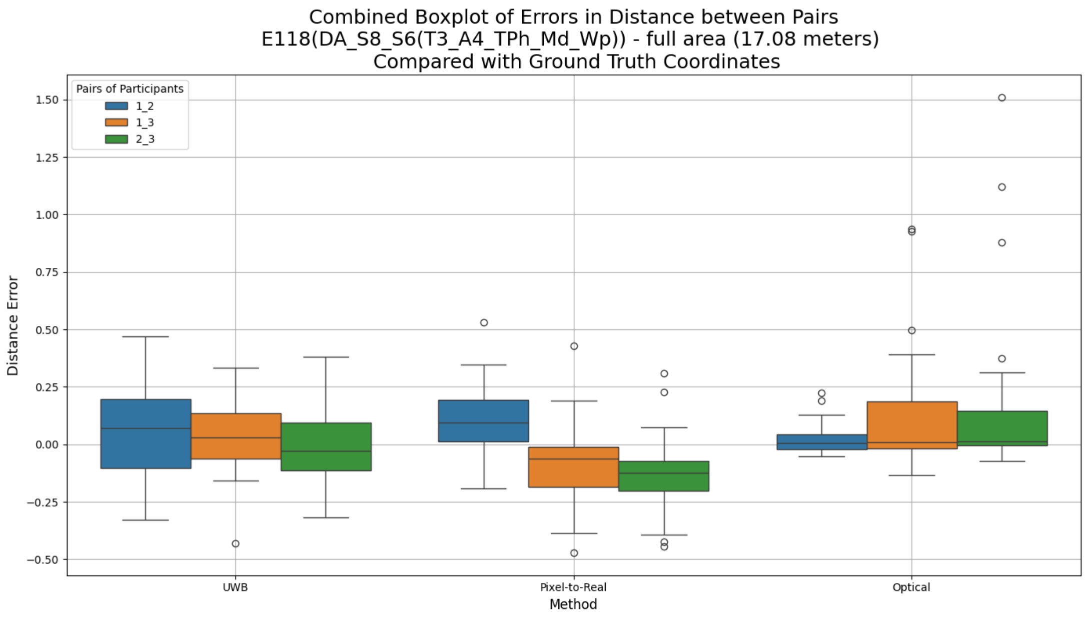


Figure A.28 The box plot illustrates the distribution of errors in distance between people estimated by UWB, Pixel-to-Real and Optical methods compared to the Ground Truth distances between people. This data is collected in $\texttt{E}_{118}$, considering the entire area of the environment, which is 17.08 meters long. It is important to note that this plot contains negative as well as positive distance error values.

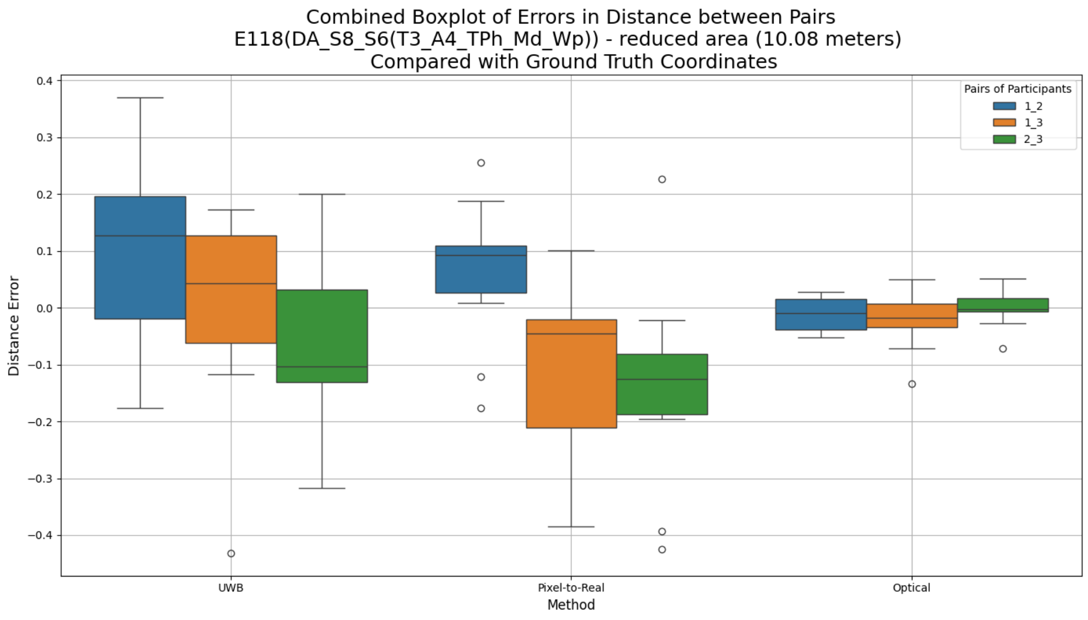


Figure A.29 The box plot illustrates the distribution of errors in distance between people estimated by UWB, Pixel-to-Real and Optical methods compared to the Ground Truth distances between people. This data is collected in $\mathtt{E}_{118}$, considering the reduced area of the environment, which is 10.08 meters long. It is important to note that this plot contains negative as well as positive distance error values.

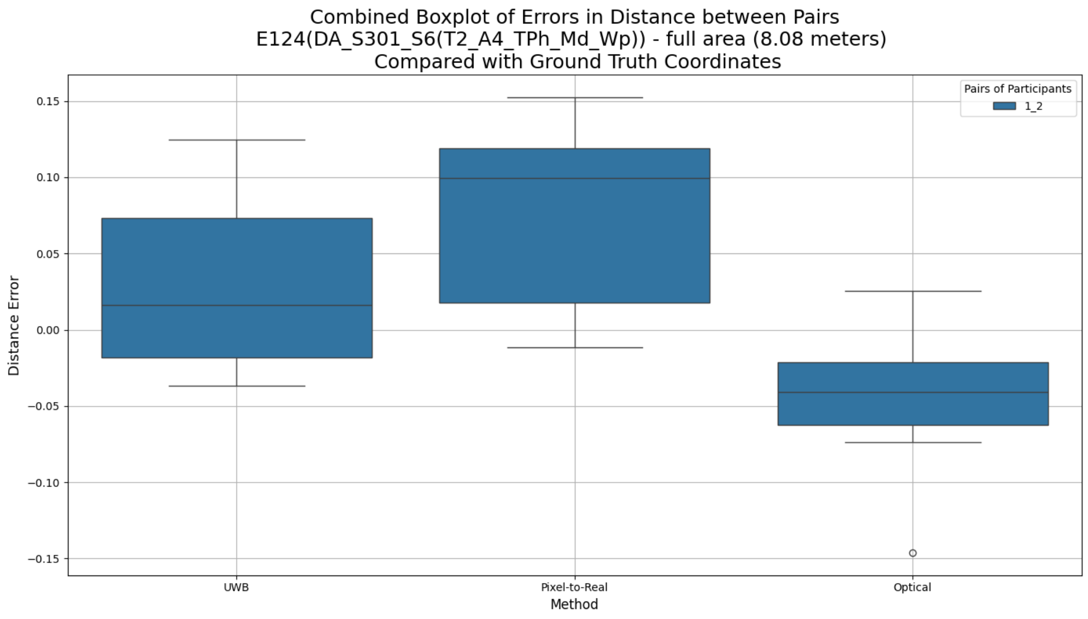


Figure A.30 The box plot illustrates the distribution of errors in distance between people estimated by UWB, Pixel-to-Real and Optical methods compared to the Ground Truth distances between people. This data is collected in $\mathtt{E}_{124}$, considering the entire area of the environment, which is 8.08 meters long. It is important to note that this plot contains negative as well as positive distance error values.

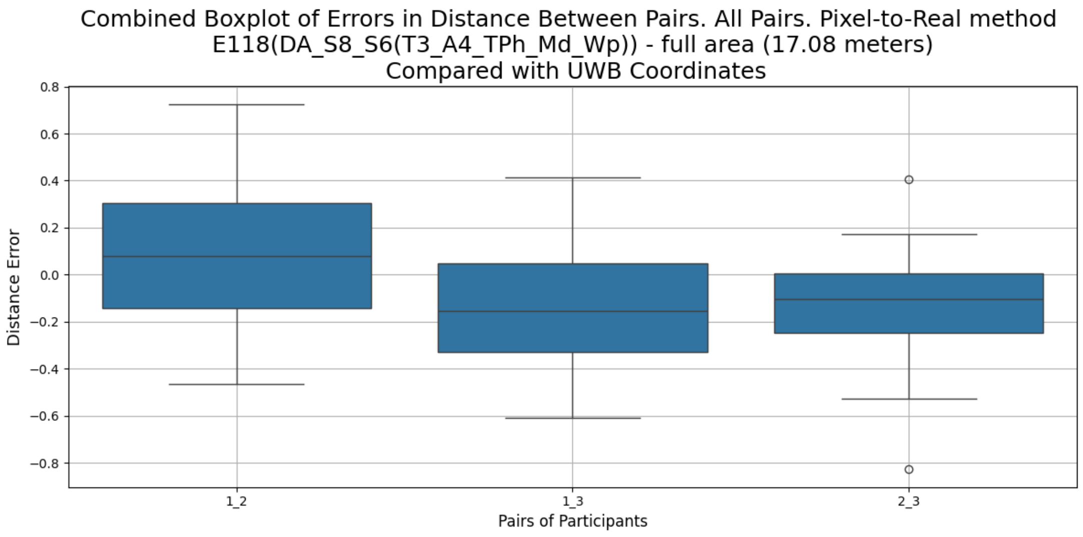


Figure A.31 The box plot illustrates the distribution of errors in distance between people estimated by Pixel-to-Real method compared to the distances between people estimated by UWB method. This data is collected in $\mathtt{E}_{118}$, considering the entire area of the environment, which is 17.08 meters long. It is important to note that this plot contains negative as well as positive distance error values.

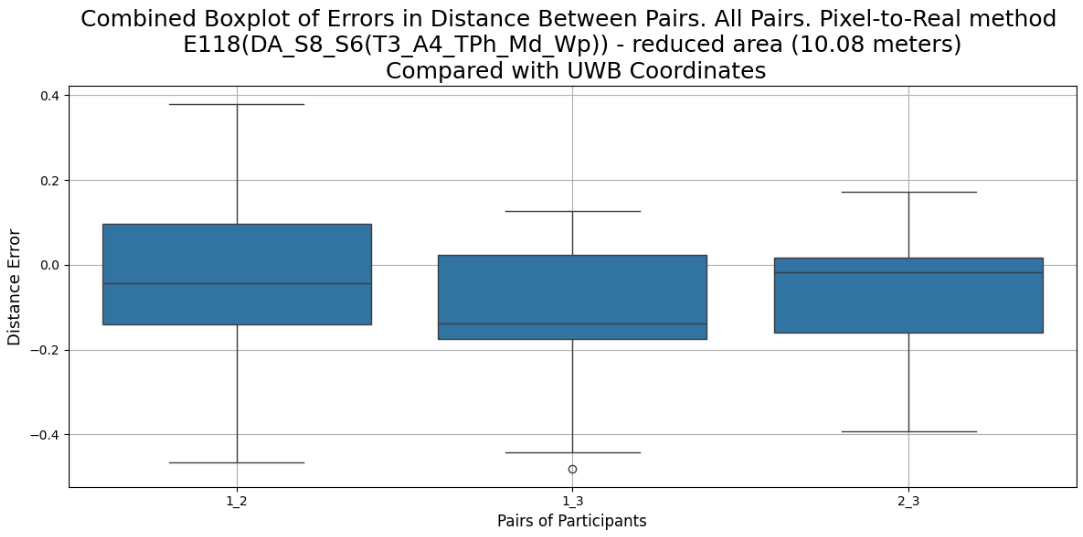


Figure A.32 The box plot illustrates the distribution of errors in distance between people estimated by Pixel-to-Real method compared to the distances between people estimated by UWB method. This data is collected in $\mathtt{E}_{118}$, considering the reduced area of the environment, which is 10.08 meters long. It is important to note that this plot contains negative as well as positive distance error values.

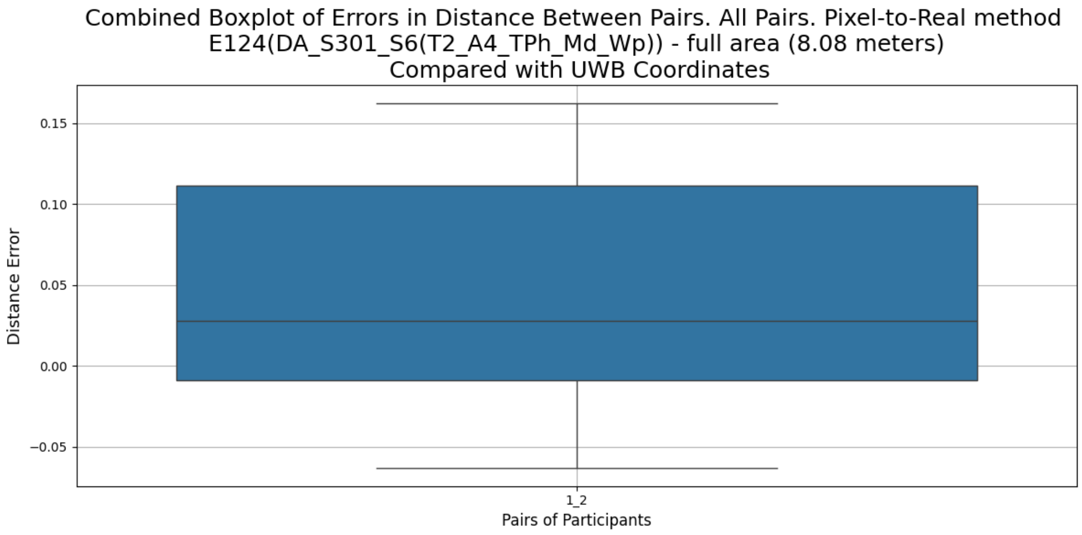


Figure A.33 The box plot illustrates the distribution of errors in distance between people estimated by Pixel-to-Real method compared to the distances between people estimated by UWB method. This data is collected in $\mathtt{E}_{124}$, considering the entire area of the environment, which is 8.08 meters long. It is important to note that this plot contains negative as well as positive distance error values.

Bibliography

1. BREZANI, S. et al. Smart extensions to regular cameras in the industrial environment. *Procedia Computer Science*. 2022, vol. 200, pp. 298–307.
2. LI, Q. et al. Fingerprint and Assistant Nodes Based Wi-Fi Localization in Complex Indoor Environment. *IEEE Access*. 2016, vol. 4, pp. 2993–3004. Available from DOI: `10.1109/ACCESS.2016.2579879`.
3. DICKINSON, P. et al. Indoor positioning of shoppers using a network of Bluetooth Low Energy beacons. In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2016, pp. 1–8.
4. LIU, M. et al. RFID 3-D indoor localization for tag and tag-free target based on interference. *IEEE Transactions on Instrumentation and Measurement*. 2018, vol. 68, no. 10, pp. 3718–3732.
5. SILVA, B. et al. Experimental study of UWB-based high precision localization for industrial applications. In: *2014 IEEE International Conference on Ultra-WideBand (ICUWB)*. 2014, pp. 280–285. Available from DOI: `10.1109/ICUWB.2014.6958993`.
6. YANG, Z. et al. Bird's-eye View Social Distancing Analysis System. In: *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*. 2022, pp. 427–432. Available from DOI: `10.1109/ICCWorkshops53468.2022.9814627`.
7. ZHANG, W. et al. Deep neural networks for wireless localization in indoor and outdoor environments. *Neurocomputing*. 2016, vol. 194, pp. 279–287.
8. ZAFARI, F. et al. A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials*. 2019, vol. 21, no. 3, pp. 2568–2599.
9. WIKIPEDIA CONTRIBUTORS. *Received signal strength indicator — Wikipedia, The Free Encyclopedia* [`https://en.wikipedia.org/w/index.php?title=Received_signal_strength_indicator&oldid=1226746544`]. 2024. [Online; accessed 8-July-2024].
10. LINDHÉ, M. et al. An experimental study of exploiting multipath fading for robot communications. In: *Robotics: Science and Systems*. 2007, p. 49.
11. WIKIPEDIA CONTRIBUTORS. *ISM radio band — Wikipedia, The Free Encyclopedia* [`https://en.wikipedia.org/w/index.php?title=ISM_radio_band&oldid=1218327506`]. 2024. [Online; accessed 14-April-2024].
12. WIKIPEDIA CONTRIBUTORS. *Bluetooth Low Energy — Wikipedia, The Free Encyclopedia* [`https://en.wikipedia.org/w/index.php?title=Bluetooth_Low_Energy&oldid=1218377189`]. 2024. [Online; accessed 14-April-2024].
13. DHIEB, M. et al. A Gaussian pulse generator for ultra-wideband radar system. In: *Proc. Septiémes Journées Scientifiques des Jeunes Chercheurs en Génie Electrique et Informatique (GEI)*. 2007, pp. 563–573.

14. Botler, L. et al. Direction Finding with UWB and BLE: A Comparative Study. In: *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. 2020, pp. 44–52. Available from DOI: 10.1109/MASS50613.2020.00016.

15. Bahadori, S. et al. Real-time people localization and tracking through fixed stereo vision. *Applied Intelligence*. 2007, vol. 26, pp. 83–97.

16. Wikipedia contributors. *Stereopsis — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/w/index.php?title=Stereopsis&oldid=1220871393]. 2024. [Online; accessed 8-July-2024].

17. Qorvo. *TECHNOLOGY* [https://www.qorvo.com/innovation/ultra-wideband/technology]. 2018. [Online; accessed 26-April-2024].

18. Qorvo. *APS014 APPLICATION NOTE. ANTENNA DELAY CALIBRATION OF DW1000-BASED PRODUCTS AND SYSTEMS* [https://www.qorvo.com/products/d/da008449]. 2018. [Online; accessed 16-April-2024].

19. Wikipedia contributors. *Clock drift — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/w/index.php?title=Clock_drift&oldid=1210473399]. 2024. [Online; accessed 27-April-2024].

20. Kwak, M.; Chong, J. A new Double Two-Way Ranging algorithm for ranging system. 2010. Available from DOI: 10.1109/ICNIDC.2010.5657814.

21. Dardari, D. et al. Ranging With Ultrawide Bandwidth Signals in Multipath Environments. *Proceedings of the IEEE*. 2009, vol. 97, pp. 404–426. Available from DOI: 10.1109/JPROC.2008.2008846.

22. Neirynck, D. et al. An alternative double-sided two-way ranging method. In: *2016 13th Workshop on Positioning, Navigation and Communications (WPNC)*. 2016, pp. 1–4. Available from DOI: 10.1109/WPNC.2016.7822844.

23. Qorvo. *Implementation of Two-Way Ranging with the DW1000 (Application Note APS013)* [https://www.qorvo.com/products/d/da008448]. 2015. [Online; accessed 29-April-2024].

24. Jeon, S. *ToA based positioning with DW1000 UWB module* [https://github.com/somidad/dw1000-positioning]. GitHub, 2018.

25. Piotr. *Multiple Tags - best way to achieve it* [https://github.com/thotro/arduino-dw1000/issues/181]. GitHub, 2017. [Online; accessed 27-April-2024].

26. Wikipedia contributors. *Triangulation — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/w/index.php?title=Triangulation&oldid=1217396660]. 2024. [Online; accessed 27-April-2024].

27. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. KDD '16. ISBN 9781450342322. Available from DOI: 10.1145/2939672.2939785.

28. Wikipedia contributors. *Planar projection — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/w/index.php?title=Planar_projection&oldid=1142967567]. 2023. [Online; accessed 13-July-2024].

29. Hata, K.; Savarese, S. *CS231A Course Notes 1: Camera Models* [`https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf`]. Stanford University, 2015. [Online; accessed 21-May-2024].

30. Wikipedia contributors. *Camera resectioning — Wikipedia, The Free Encyclopedia* [`https://en.wikipedia.org/w/index.php?title=Camera_resectioning&oldid=1219295408`]. 2024. [Online; accessed 21-May-2024].

31. Mazhar, F. et al. Precise Indoor Positioning Using UWB: A Review of Methods, Algorithms and Implementations. *Wireless Personal Communications*. 2017, vol. 97, pp. 4467–4491.

List of Figures

List of Tables

List of Abbreviations

$\mathtt{E}_{109}$ This experiment represents a data acquisition `DA` conducted in the `S8` environment; it follows the $\mathtt{S}_5$ scenario and involves the use of 1 tag and 2 anchors; tags are placed on hands ($\mathtt{TP}_\mathtt{h}$); the experiment involves dynamic movement ($\mathtt{M}_\mathtt{d}$) with walking pattern both on predefined lines and next to them ($\mathtt{W}_\mathtt{pn}$); it produces $\mathtt{D}_\mathtt{raw}\mathtt{(E}_{109}\mathtt{)}$ set of distances.
$\mathtt{E}_{109}\mathtt{(DA\_S8\_S}_5\mathtt{(T1\_A2\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{pn}\mathtt{))} \rightarrow \mathtt{D}_\mathtt{raw}\mathtt{(E}_{109}\mathtt{)}$. 59–62, 64–73, 77, 117–119

$\mathtt{E}_{113}$ This experiment represents a data acquisition `DA` conducted in the `S301` environment; it follows the $\mathtt{S}_5$ scenario and involves the use of 1 tag and 2 anchors; tags are placed on hands ($\mathtt{TP}_\mathtt{h}$); the experiment involves dynamic movement ($\mathtt{M}_\mathtt{d}$) with walking pattern both on predefined lines and next to them ($\mathtt{W}_\mathtt{pn}$); it produces $\mathtt{D}_\mathtt{raw}\mathtt{(E}_{113}\mathtt{)}$ set of distances.
$\mathtt{E}_{113}\mathtt{(DA\_S301\_S}_5\mathtt{(T1\_A2\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{pn}\mathtt{))} \rightarrow \mathtt{D}_\mathtt{raw}\mathtt{(E}_{113}\mathtt{)}$. 59–62, 65–67, 70–73, 117–119

$\mathtt{E}_{118}$ This experiment represents a data acquisition `DA` conducted in the `S8` environment; it follows the $\mathtt{S}_6$ scenario and involves the use of 3 tags and 4 anchors; tags are placed on hands ($\mathtt{TP}_\mathtt{h}$); the experiment involves dynamic movement ($\mathtt{M}_\mathtt{d}$) with walking pattern on predefined lines only ($\mathtt{W}_\mathtt{p}$); it produces $\mathtt{D}_\mathtt{raw}\mathtt{(E}_{118}\mathtt{)}$ set of distances.
$\mathtt{E}_{118}\mathtt{(DA\_S8\_S}_6\mathtt{(T3\_A4\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{p}\mathtt{))} \rightarrow \mathtt{D}_\mathtt{raw}\mathtt{(E}_{118}\mathtt{)}$. 59–63, 65–70, 74, 75, 77–79, 117–119

$\mathtt{E}_{124}$ This experiment represents a data acquisition `DA` conducted in the `S301` environment; it follows the $\mathtt{S}_6$ scenario and involves the use of 2 tags and 4 anchors; tags are placed on hands ($\mathtt{TP}_\mathtt{h}$); the experiment involves dynamic movement ($\mathtt{M}_\mathtt{d}$) with walking pattern on predefined lines only ($\mathtt{W}_\mathtt{p}$); it produces $\mathtt{D}_\mathtt{raw}\mathtt{(E}_{124}\mathtt{)}$ set of distances.
$\mathtt{E}_{124}\mathtt{(DA\_S301\_S}_6\mathtt{(T2\_A4\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{p}\mathtt{))} \rightarrow \mathtt{D}_\mathtt{raw}\mathtt{(E}_{124}\mathtt{)}$. 59, 60, 62, 64–68, 70, 74–79, 117–119

$\mathtt{E}_{5}$ This experiment represents a data acquisition `DA` conducted in the `S301` environment; it follows the $\mathtt{S}_6$ scenario and involves the use of 3 tags and 2 anchors; tags are placed on hands ($\mathtt{TP}_\mathtt{h}$); the experiment involves dynamic movement ($\mathtt{M}_\mathtt{d}$) with walking pattern on predefined lines only ($\mathtt{W}_\mathtt{p}$); it produces $\mathtt{D}_\mathtt{raw}\mathtt{(E}_{5}\mathtt{)}$ set of distances.
$\mathtt{E}_{5}\mathtt{(DA\_S301\_S}_6\mathtt{(T3\_A2\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{d}\mathtt{\_W}_\mathtt{p}\mathtt{))} \rightarrow \mathtt{D}_\mathtt{raw}\mathtt{(E}_{5}\mathtt{)}$. 17, 18, 111, 117

$\mathtt{E}_{6}$ This experiment represents an UWB calibration `Calib` conducted in the `S301` environment; it follows the $\mathtt{S}_4$ scenario and involves the use of 1 tag and 2 anchors; tags are placed on hands ($\mathtt{TP}_\mathtt{h}$); the experiment is static ($\mathtt{M}_\mathtt{s}$); it produces $\mathtt{D}_\mathtt{raw}\mathtt{(E}_{6}\mathtt{)}$ set of distances.
$\mathtt{E}_{6}\mathtt{(Calib\_S301\_S}_4\mathtt{(T1\_A2\_TP}_\mathtt{h}\mathtt{\_M}_\mathtt{s}\mathtt{\_W}_\mathtt{p}\mathtt{))} \rightarrow \mathtt{D}_\mathtt{raw}\mathtt{(E}_{6}\mathtt{)}$. 40

$\mathtt{E}_{83}$ This experiment represents an UWB calibration `Calib` conducted in the `Dorm` environment; it follows the $\mathtt{S}_3$ scenario and involves the use of 3 tags and 2 anchors; tags are placed on plastic pipes ($\mathtt{TP}_\mathtt{p}$); the experiment involves dynamic movement ($\mathtt{M}_\mathtt{d}$) with walking pattern on predefined lines only ($\mathtt{W}_\mathtt{p}$);

it produces $\mathtt{D_{raw}(E_{83})}$ set of distances.
$\mathtt{E_{83}(Calib\_Dorm\_S_3(T3\_A2\_TP_p\_M_d\_W_p))} \rightarrow \mathtt{D_{raw}(E_{83})}$. 39, 40, 112

BLE Bluetooth Low Energy. 5, 6

BLE v5.1 Bluetooth Low Energy v5.1. 6

CCTV Closed-circuit television. 1, 3, 6

CNN Convolutional Neural Network. 49

Covid-19 Coronavirus disease 2019. 1, 73

Dorm - ExpEnv1 Dormitory - Experiment Environment 1. 34, 35, 82, 83, 112

DS-TWR Double-Sided Two-Way Ranging. 23, 24

FoV Field of View. 54

GUI Graphical User Interface. 2

Industry 4.0 Fourth Industrial Revolution. 2

IoT Internet of Things. 2, 7, 19

LoS Line of Sight. 6, 11, 14, 20, 40, 56, 111

MAE Mean Absolute Error. 62, 63, 71, 72, 74–77, 79

MQTT MQ Telemetry Transport. 19

MSE Mean Squared Error. 64

NLoS Non-line of Sight. 5, 11, 12, 35, 58, 111

OWR One-Way Ranging. 21

PC Personal Computer. 1

RFID Radio Frequency Identification Device. 5

Rot - ExpEnv2 Rotunda - Experiment Environment 2. 34, 36, 82, 112

RSSI Received Signal Strength Indicator. 5

S301 - ExpEnv3 S301 - Experiment Environment 3. 34, 36, 59, 60, 63, 66, 72, 73, 79, 82, 112

S8 - ExpEnv4 S8 - Experiment Environment 4. 34, 37, 59, 60, 62, 65, 66, 71–73, 79, 82, 84, 112

SoS System-on-a-Chip. 8

TDoA	Time Difference of Arrival. 21
ToF	Time of Flight. 6, 9, 21–23
TWR	Two-Way Ranging. 20, 21
UWB	Ultra-wideband. 2, 5, 6
XGBoost	Extreme Gradient Boosting. 52
YOLOv4	You Only Look Once version 4. 49, 50, 53

A Attachments

A.1 Source code (Implementation)

- **Indoor Positioning System** the main GUI for data visualization and analysis
- **ESP32 UWB** the firmware for ESP32 UWB devices
- **Server** the multi-threaded server for video and UWB data recording
- **PixelToReal** training of the Pixel-to-Real model
- **Camera Intrinsic Calibration** the calibration of the camera for Optical method

A.2 Data for Indoor Positioning System (GUI)

The data prepared to open in GUI.

A.3 Archive

Contains a journal for notes collected during experiments and project development and papers used as a motivation for the work.

A.4 GitHub

All the data collected during experiments, source codes, results of the evaluation of the localization methods (also those, which are not included in Attachments) are available in the main repository of the Thesis on GitHub:
https://github.com/Razyapoo/Master-Thesis/

Technical and user documentations are available in the main repository of the Thesis:
https://github.com/Razyapoo/Master-Thesis/tree/main/Archive/Documentation