

Institute of Formal Methods in Computer Science

University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Bachelorarbeit

**Self-Localization of IoT Devices -
Development and Implementation of
a System for Self-Localization of
Embedded Devices based on Wi-Fi
Information**

Alexander Schäfer

Course of Study: Softwaretechnik

Examiner: Prof. Dr. Stefan Funke

Supervisor: Dr. Daniel Bahrtd

Commenced: January 25, 2022

Completed: July 25, 2022

Abstract

Smart homes have become a reoccurring topic in our society. Many homes have been modernized in such a way that technology takes a prevailing role. As a result, many electronics technicians have to adapt to the ongoing modernization. With that said, many improvements can be implemented in order for technicians to work more efficiently and flawlessly. One of these improvements could be a localization software that locates IoT devices in a household. As many IoT devices are installed in walls, the respective location cannot be determined without a great margin of error. A software that locates these devices and maps them to the respective household can prove to be very beneficial. In order for such a software to exist we need to determine the distance between IoT devices through some mean. Wi-fi signals consist of Channel State Information (CSI) that could possibly describe the physical distance between devices.

This thesis focuses on the evaluation of Channel State Information in Wi-Fi signals and the possibility of using this information to determine the distance of devices to each other. If the CSI information proves to be successful in determining the distance, a complete software will be implemented that calculates the positions of all IoT devices in a local network. For this thesis we will focus on ESP32 microcontrollers that already utilize CSI data for human movement detection.

Kurzfassung

Smart Homes sind zu einem wiederkehrenden Thema in unserer Gesellschaft geworden. Viele Häuser wurden so modernisiert, dass die Technik eine dominierende Rolle einnimmt. Dadurch müssen sich viele Elektroniker auf die laufende Modernisierung einstellen. Vor diesem Hintergrund können viele Verbesserungen implementiert werden, damit die Techniker effizienter und fehlerfreier arbeiten können. Eine dieser Verbesserungen könnte eine Lokalisierungssoftware sein, die IoT-Geräte in einem Haushalt ortet. Da viele IoT-Geräte in Wänden verbaut sind, kann der jeweilige Standort nicht ohne hohe Fehlerquote bestimmt werden. Eine Software, die diese Geräte ortet und dem jeweiligen Haushalt zuordnet, kann sich als sehr vorteilhaft erweisen. Damit eine solche Software existieren kann, müssen wir die Entfernung zwischen IoT-Geräten bestimmen. WLAN-Signale bestehen aus Channel State Informations (CSI), die möglicherweise die physische Entfernung zwischen Geräten beschreiben könnten.

Diese Arbeit konzentriert sich auf die Auswertung von Channel State Information in WLAN-Signalen und die Möglichkeit diese Informationen zu verwenden, um die Entfernung von Geräten zueinander zu bestimmen. Erweisen sich die CSI-Informationen bei der Entfernungsbestimmung als erfolgreich, wird eine komplette Software implementiert, die die Positionen aller IoT-Geräte in einem lokalen Netzwerk berechnet. In dieser Arbeit konzentrieren wir uns auf ESP32-Mikrocontroller, die bereits CSI-Daten zur Erkennung menschlicher Bewegungen verwenden.

Contents

1	Introduction	17
1.1	Related Work	18
2	Background	21
2.1	Channel State Information (CSI)	21
2.2	Espressif	24
2.3	ZeroMQ	26
2.4	A-Frame	27
3	Approach	29
4	Implementation	31
4.1	Localization Algorithms	31
4.2	Network Protocols	36
5	Experiments and Results	39
5.1	Test environments for short distance measurements	39
5.2	Test environments for long distance measurements	41
5.3	Problems and Challenges	42
5.4	Findings	43
5.5	Application - Radar Localization	51
6	Conclusion and Future Work	53
6.1	Future Work	54
	Bibliography	57
A	Mathematical prerequisites	59

List of Figures

2.1	OFDM and FDM Example	22
2.2	Chipmodule - Wrover-E	25
2.3	Front and Back of the ESP32 Devkits(-R) Board	25
2.4	Example: Outcome of the Code in A-Frame	28
4.1	GPS Example	32
4.2	Network protocol for the experiment	36
5.1	Layout for Environments 1-3 in a private household	40
5.2	Layout for Environments 4-6 located on the first floor in U38	41
5.3	Layout for Environment 7 in U38 basement	42
5.4	Comparison of RSSI and distance for Environments 1-3	43
5.5	Comparison of STD and CORR with distance for Environments 1-3	44
5.6	Comparison of CORR and STD for Environments 1-3	45
5.7	Comparison of STD and CORR with RSSI for Environments 1-3	45
5.8	Comparison of STD and CORR with RSSI and distance for Environments 1-3	46
5.9	Comparison of RSSI and distance for Environments 4-7	47
5.10	Comparison of STD and CORR with distance for Environments 4-7	48
5.11	Comparison of CORR and STD for Environments 4-7	49
5.12	Comparison of STD and CORR with RSSI for Environments 4-7	49
5.13	Comparison of STD and CORR with RSSI and distance for Environments 4-7	50
5.14	Radar-App	51

List of Tables

2.1	RSSI Distribution	23
-----	-----------------------------	----

List of Listings

2.1	Pyzmq - Python-Code	26
2.2	A-Frame - HTML-Code	27
2.3	A-Frame - JavaScript-Code	28

List of Algorithms

4.1	GPS-oriented algorithm	34
A.1	Sphere-Sphere Intersection	59
A.2	Circle-Sphere Intersection	60
A.3	PointPair-Sphere Intersection	61

Acronyms

- AP** Access Point. 17
- CORR** Correlation Coefficient between Subcarriers. 23
- CSI** Channel State Information. 17
- FDM** Frequency Division Multiplexing. 22
- GPS** Global Positioning Systems. 17
- IoT** Internet of Things. 17
- KNN** K-Nearest Neighbour. 18
- LQI** Link Quality Indicator. 18
- OFDM** Orthogonal Frequency-Division Multiplexing. 21
- RSSI** Received Signal Strength Indicator. 17
- SDP** Semi-Definite Programming. 35
- SNLP** Sensor Network Localization Problem. 31
- SOCP** Second Order Cone Programming. 35
- STA** Station. 17
- STD** Covariance between Subcarriers. 23
- TCP** Transmission Control Protocol. 24
- TOF** Time-of-Flight. 18
- UDP** User Datagram Protocol. 24

1 Introduction

In today's society Global Positioning Systems (GPS) are used in a variety of ways. It is not only limited to tracking oneself but, such is the case for iPhone users, to track phones that have gone missing. As GPS is a valuable technology used to track certain electronic devices around the globe, one cannot dismiss the fact that the margin for error can have an impact on the exact location of the given devices. It is, for example, not optimal to use GPS tracking for many devices in a small indoor area. Many homes and warehouses encompass Internet of Things (IoT) devices that are connected in a local network of which the exact location can be beneficial for commercial and non-commercial use. Electricians could use a localization of these IoT devices to find specific devices installed in walls and other objects. As a result, the question arises whether the data of Wi-Fi signals can be used as a distance measurement to evaluate and determine the positioning of all connected devices in a local network.

An experiment has been formed in chapter 5 in order to determine the distance between two devices through specific variables in Wi-Fi signals. All IoT devices use Received Signal Strength Indicator (RSSI) to determine the strength of Wi-Fi signals between two nodes. Additionally some devices support Channel State Information (CSI). Both have been evaluated to find a pattern at specific distances to determine the physical distance of devices. Espressif's ESP32 boards support CSI and can create their own local network in which Wi-Fi signals can be measured. Therefore ESP32 boards, more specifically ESP32-DevKitS(-R) boards, have been used as Access Point (AP) and as Station (STA) to access CSI data and to create a closed local network. Computers and smartphones have also been used as a mean to receive and evaluate CSI. These measurements are then processed to reliably estimate distances. The distance measurements are generally performed in an indoor environment in order to determine the viability of the localization with physical objects possibly obstructing the signal. The main reason for such an environment is to mimic an area that generally occurs in homes and Smart Homes. A perfect environment, where no object or signal obstructs the given Wi-Fi signal, would not be beneficial and meaningful for commercial use, as this is rarely the case and does not correlate with normally occurring indoor areas.

If the distance estimation proves to be successful, reliable and efficient, an algorithm in 4.1 is used to determine the positions of all the devices in a 3D environment with the input only being the distances of the devices to each other. This projection of the devices in the 3D environment is to be realized in augmented reality in order for the location of the devices to be visible through a smartphone camera. For all distances to be determined, an efficient network protocol needs to be implemented that outputs the distances between all devices without possible interferences and manipulations of the RSSI and CSI values. These protocols are explained more in depth in 4.2. The results of the experiment are fully explained in 5.4.1 and 5.4.2.

The requirements presented have been set in such a way that the resulting product, if distances can be estimated correctly, can be used in commercial use. This includes an efficient and error-tolerant

evaluation of the data in Wi-Fi signals with a correct physical distance estimation and a product that is user friendly and easily usable without the need for basic knowledge in computer networks. The complete project with all its software components can be viewed on GitHub¹. All the data gathered in the experiment can also be viewed in the previously mentioned github project. All figures contained in the thesis that are not made by us have been footmarked with their link being shown before the bibliography under "Notes". The footmarks of figures are represented as negative integers in order to distinguish them from other footmarks with their link being depicted on the same page.

1.1 Related Work

K. Benkič, M. Malajner, P. Planinšič and Ž. Čučej created an experiment using RSSI values for the estimation of distances presented in their scientific paper [BMPC08]. As RSSI is very sensitive to scattering, reflection and other physical properties and thus have a great impact on RSSI values, a second value has been used to gain a better signal strength indication for distance estimation. They used Zigbee-based modules to extract RSSI and Link Quality Indicator (LQI) values to determine distances. LQI estimated how easily the received signal can be modulated while taking noise in the channel into consideration. As a result, they were able to extract the strength of a signal more precisely by using a combination of RSSI and LQI values. The experiment concluded that with enough packets sent, the distance can be precisely evaluated when terrain variables and conditions are met. These conditions needed to be nearly perfect for a successful estimation and are thus not optimal for practical use.

Lorenz Schauer, Florian Dorfmeister and Marco Maier performed an experiment using Time-of-Flight (TOF)) to determine Wi-Fi positionings in [SDM13]. Their approach focused on a more simpler variable that is less influenced by external events and does not need any additional calibration. The experiment was formed by estimating Round-Trip Time-of-Flight (RTOF) with NULL-ACK-Sequences. They had to therefore measure the Round-Trip-Time (RTT) to receive the given information. The paper concluded that "the accuracy of such systems is limited due to missing precise and adequate hardware timers". Additionally, their evaluation showed that the mean deviation of all distances measured is always greater than 6.26 meters. Thus, the presented ranging approach is not accurate enough for indoor environments and for general applications.

David Sánchez-Rodríguez, Miguel A. Quintana-Suárez and Itziar Alonso-González, Carlos Ley-Bosch and Javier J. Sánchez-Medina created an experiment in which they used both CSI and RSSI and presented it in their paper "Fusion of Channel State Information and Received Signal Strength for Indoor Localization Using a Single Access Point" in [SQA+20]. The experiment focused on collecting RSSI and CSI while processing the amplitudes of CSI. These two values have then been fused by concatenation and put in a fingerprint dataset. Afterwards, the datasets have been evaluated and trained using several algorithms such as K-Nearest Neighbour (KNN) and Bayesian classification. KNN proved to be very efficient with an error distance of at least 0.15 meters in certain environments. The downside of this experiment is the training necessary for a

¹<https://github.com/classicABCD/NetworkDeviceLocalization>

machine learning algorithm. As the environment still plays a role on how the data is perceived, the algorithm needs to learn the new dataset to achieve the highest accuracy. As for commercial use, this paper proves to be the most efficient and accurate approach for distance estimation.

2 Background

This chapter focuses on the technologies and frameworks used for the experiment and the development of the applications.

2.1 Channel State Information (CSI)

Wireless Channel State Information (CSI) refers to channel properties in wireless communications, as explained in [MZW19]. They describe how signals propagate from the transmitter to the receiver and represent several combined effects, such as scattering, power decay and fading with increased distance.

Scattering describes physical processes where moving particles or radiation deviate from a straight trajectory by localized non-uniformities. This includes reflected radiation which can, in our use-case, greatly affect Wi-Fi signals, depending on the type of material surrounding the nodes. Fading is a variation of the dampening or attenuation of a signal with numerous variables, such as time, geographical position and radio frequency.

CSI is extremely sensitive to environmental changes. These environmental changes cannot only be caused by general movements such as walking, moving and running of humans and animals but also subtle movements that are caused by breathing and other similar phenomena.

Orthogonal Frequency-Division Multiplexing (OFDM), more specifically MIMO-OFDM [Bol06], is the medium that refracts the received signal strength into CSI.

2.1.1 MIMO-OFDM

MIMO-OFDM is a combination of two principles, Multiple Input Multiple Output (MIMO) and Orthogonal Frequency-Division Multiplexing (OFDM). These terms are explained further in the next section.

Multiple Input Multiple Output (MIMO)

Multiple Input Multiple Output (MIMO) is a term applied for processes that improve wireless connections with multiple antennas used in parallel. This results in several antennas delivering a better reception signal and increasing the possible distance and overall data throughput.

MIMO refers to the ability of radio systems to send and receive a data stream simultaneously via multiple antennas through the use of frequency, time, space and intelligent signal processing. Each antenna receives the radio signals from all transmitting antennas on the opposite side and calculate an optimal input signal, taking the changing properties of the radio channel into account.

Frequency Division Multiplexing (FDM)

In Frequency Division Multiplexing (FDM) a wide frequency band is divided into several narrow frequency bands. Each individual frequency band has a carrier frequency. Furthermore, a single data channel is assigned to each of these carrier frequencies. The data is transmitted simultaneously and independently of each other to prevent overlapping of signals. The modulated signals are then combined on the transmit side using a multiplexer. The combined signal is transmitted via the communication channel so that several independent data streams can be transmitted simultaneously. On the receiving side, the individual signals are extracted from the combined signal by demultiplexing (DEMUX). The figure in 2.1 depicts how the signals are combine through a multiplexer.

Orthogonal Frequency-Division Multiplexing (OFDM)

Orthogonal Frequency Division Multiplexing is a variation of FDM and is a method of digital data being encoded on multiple carrier frequencies. It is used for many of the latest wireless and telecommunication standards such as Wi-Fi 802.11ac, 4G and 5G cellular phone technologies, satellites and many others.

OFDM allows multiple users to share one link by dividing available bandwidth into different overlapping sub-channels that are closely spaced. Thus, OFDM would allow more data transmissions than FDM. In order for interference to be prevented in the overlapping sub-channels, signals in a channel would be combined in a way that they are orthogonal to each other. As a result, all signals in a channel operate without dependence and interference.

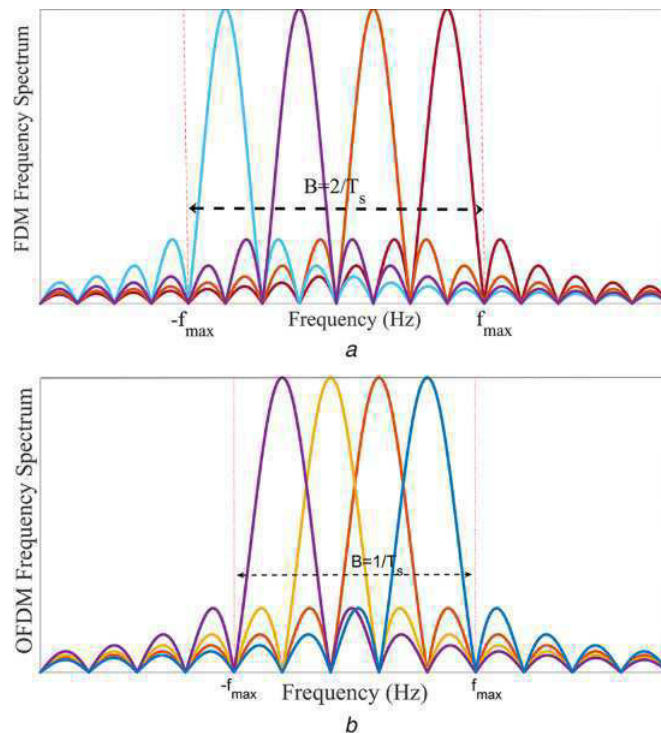


Figure 2.1: OFDM and FDM Example [GD17]

The figure in 2.1 depicts how signals are combined in FDM and OFDM. B denotes the bandwidth used and T the duration of a signal. In the figure it is apparent that OFDM uses half of the bandwidth in comparison to FDM. FDM uses a bandwidth of $\frac{2}{T}$ while OFDM a bandwidth of only $\frac{1}{T}$. Additionally the orthogonality in OFDM is achieved by multiplexing the signals in such a way that the peak of one signal occurs at the null of all other neighbouring signals.

2.1.2 Relevant CSI values

In the Espressif Framework, more specifically esp-csi (2.2.1), RSSI is combined with CSI, even though RSSI is generally not considered as a value in CSI. The CSI values are derived from the RSSI values with the utilization of MIMO-OFDM. The values in esp-csi, that are used for a possible distance estimation, are the Covariance between Subcarriers (STD) and the Correlation Coefficient between Subcarriers (CORR).

Received Signal Strength Indicator

The RSSI is a measurement that represents the power present in a received radio signal and the quality of a received signal which in most cases indicates how well a particular radio can hear remote connected client radios. At larger distances, the signal gets weaker and the wireless data rates get slower leading to an overall lower data throughput. RSSI is not represented in a subset of CSI, as it is calculated and used independently of it. Therefore, RSSI has to be seen as an own form for the measurement of distances in this experiment. RSSI can take values of 0 up to 255 depending on the chipset manufacturer. As a result, RSSI is often used with received signal power due to received signal power being measured in decibels or decibel-milliwatts (dBm) on a logarithmic scale. This measurement is uniform on all chipsets and is thus a main measurement for signal strength. For our experiment, we will use the dBm representation of RSSI. A RSSI value r is restricted to the interval $r \in [-100dBm, 0dBm]$.

The following table shows the connection strength at certain RSSI values:

Signal Strength	Connection
-40 dBm	Amazing
-67 dBm	Very Good
-70 dBm	Okay
-80 dBm	Not Good
-90 dBm	Unusable

Table 2.1: RSSI Distribution

In the table 2.1, different RSSI values are presented with their respective connectivity. As seen in 2.1, the closer the RSSI value approaches 0 dBm , the better the signal and the connection is.

Covariance between Subcarriers (STD) and Correlation Coefficient between Subcarriers (CORR)

The values STD and CORR are not explicitly explained in esp-csi and on their documentation website, more specifically under the section CSI¹. A forum entry², addressing STD and CORR, has been made for a possible explanation of the given CSI values. The forum entry has been last checked on the 24th of July 2022. The papers [GSC+18] and [WSL04] can also give possible derivations of the given values. Generally, STD's initial value is 0 while the CORR's initial value is 1 with both values being restricted to the interval [0, 1]. Human and animal movements change the CSI values depending on the severity of the movements. Subtle movements, such as breathing, can influence CSI values as well.

2.2 Espressif

Espressif Systems³ is a multinational, semiconductor company that specializes in the development of cutting-edge Wi-Fi and Bluetooth, low-power IoT Solutions. They produce cost-effective chipsets and offer open-source frameworks. Their framework is optimal for our experiment as it incorporates Wi-Fi functionalities, network protocols and the ability to exchange and extract CSI data.

2.2.1 Espressif IoT Development Framework

The Espressif IoT Development Framework⁴ (esp-idf) is intended for the development of IoT applications with Wi-Fi, Bluetooth, power management and several other system features. It is written in C and supports Python. The framework supports Espressif specific chips and devices such as ESP32. Esp-idf enables a set of network protocols such as the receiving and sending of User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) messages for both clients and servers. The framework also supports the initialization of APs and STAs for creating an own local Wi-Fi network (AP) and connecting to an existing Wi-Fi network (STA). Thus, data can be received and sent through messages when connected to the same network.

ESP-CSI

Espressif Channel State Information⁵ (esp-csi) is an application based on esp-idf and is also developed and maintained by Espressif Systems. The application uses CSI to detect human movement and other subtle phenomena such as human or animal breathing. The resulting CSI can be extracted to show the given changes in the environment. Depending on the environment layout, certain CSI values have to be adjusted to achieve correct results when trying to determine

¹<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html#wi-fi-channel-state-information>

²<https://www.esp32.com/viewtopic.php?f=13&t=28745>

³<https://www.espressif.com/en>

⁴<https://github.com/espressif/esp-idf>

⁵<https://github.com/espressif/esp-csi>

human movement. The CSI data presented in esp-csi is used in this experiment specifically for the evaluation and determination of physical distances. These distances are paired with the supporting network protocols of esp-idf to obtain the given CSI data for further evaluation.

2.2.2 ESP32

ESP32 is an inexpensive and low-power system on a chip for microcontrollers that supports a 2.4GHz band frequency, Wi-Fi and Bluetooth. The ESP32 DevkitS(-R) board in the figure 2.3 is one of the many available boards used for the deployment of an ESP32 chip.

ESP32 DevkitS(-R)

ESP32-DevKitS(-R), shown in the figure 2.3, is Espressif's flashing board designed specifically for ESP32. It can be used to flash an ESP32 module without soldering the module to the power supply and signal lines. Such a module is depicted in the figure 2.2. With a module mounted, ESP32-DevKitS(-R) can also be used as a mini development board. The ESP32-DevKitS(-R) can be powered in three ways. Either through default power supply, the micro USB port, 5V and GND header pins or 3V3 and GND header pins.

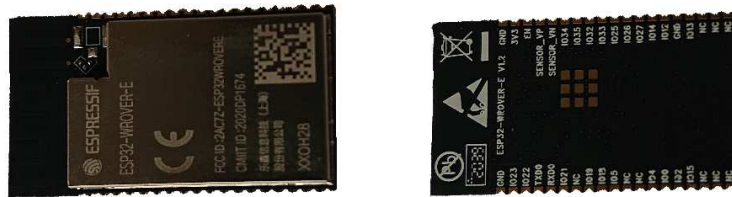


Figure 2.2: Chipmodule Front (left) and Back (right) - Wrover-E

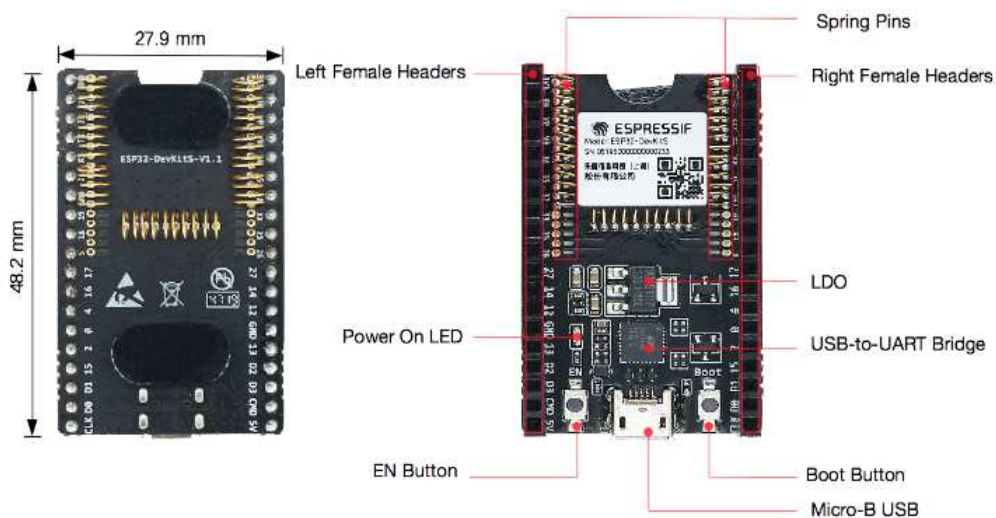


Figure 2.3: Front¹ and Back² of the ESP32 Devkits(-R) Board

2.3 ZeroMQ

ZeroMQ⁶ is an open-source universal messaging library. It enables the use of sockets that carry messages across various transport types such as UDP or TCP.

This library is implemented in several programming languages such as Java, Python or C. For our experiment we chose Python with its corresponding ZeroMQ library Pyzmq⁷. Different types of sockets can be created that can implement a client, server or a mixture of both. Clients send messages while a server waits for a message before replying. Additionally, the socket can be specified by listening to several ports and by replying to numerous messages at different ports. The following code example shows an example of Pyzmq in Python.

Listing 2.1 Pyzmq - Python-Code

```
import zmq

context = zmq.Context()
socket = context.socket(zmq.REP)
socket.bind("tcp://*:5555")

while True:
    # Wait for next request from client
    message = socket.recv()
    print("Received request: %s" % message)

    # Send reply back to client
    socket.send(b"World")
```

The Code snippet in 2.1 implements a simple *Hello World* server that waits for a message to be received and replies with a TCP message. A socket is created and stored in the *context* variable. Afterwards the socket type is initialized. In our example a *REP* socket is created that sends requests to and receives replies from exactly one service. Afterwards the socket is bound to the port 5555 and is limited to TCP messages. The last phase starts the server by waiting for a TCP message to be received and then sends a TCP reply to the given IP address via its port.

⁶<https://zeromq.org/>

⁷<https://github.com/zeromq/pyzmq>

2.4 A-Frame

A-Frame⁸ is an open-source web framework for building virtual reality experiences and is developed and maintained by Supermedium and Google. It is an entity component system framework for Three.js⁹ where developers can create 3D and WebVR scenes using HTML. In A-Frame a scene can be created with different structures such as spheres, squares and circles. These structures can be implemented either through the input of a custom HTML-component given in A-Frame or through the instantiation of components in JavaScript. In 2.2, 2.3 and 2.4 an example is shown using A-Frame with a subset of its features.

Listing 2.2 A-Frame - HTML-Code

```
<html>
  <head>
    <title>AFrame - Example</title>
    <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
  </head>

  <body>
    <a-scene id="scene" rotation="0 0 0">
      <!-- coordinate system -->
      <a-entity id="coordinate-system"></a-entity>
      <!-- spheres -->
      <a-sphere color="black" radius="0.4" position="0 0 0"></a-sphere>
      <a-sphere color="red" radius="0.4" position="0 4 4"></a-sphere>
      <!-- line -->
      <a-entity line="start: 0 0 0; end: 0 4 4" color="brown" ></a-entity>
    </a-scene>
  </body>
</html>
```

The Code in 2.2 shows that A-Frame can easily be imported through the *script*-tag without the need of downloading the framework. In the *body*-tag an *a-scene* is instantiated that creates the virtual environment for virtual reality. Several structures can be created in the scene such as *a-spheres* and *a-entities*. The *a-sphere* component creates a sphere at a given position with a set radius. The attributes of the *a-sphere* are denoted with simple HTML attributes, as seen in 2.2. *a-entities* represent complex structures when given specific attributes. In this example, two spheres are created with a radius of 0.4 and with a different color for each sphere. Their position also differs as the first sphere is positioned at (0, 0, 0) and the second sphere at (0, 4, 4). The second *a-entity* creates a line that start at the position (0, 0, 0) and ends at (0, 4, 4). The first *a-entity* with the id "coordinate-system" is created without any attributes, as its attributes are instantiated in 2.3.

⁸<https://aframe.io/>

⁹<https://threejs.org/>

Listing 2.3 A-Frame - JavaScript-Code

```
const MAX_CORD = 10;
initCoordinateSystem();

function initCoordinateSystem() {
  let system = document.getElementById("coordinate-system");
  //x-Axis
  system.setAttribute("line", `start: -${MAX_CORD} 0 0; end: ${MAX_CORD} 0 0; color: green;`);
  //y-Axis
  system.setAttribute("line__1", `start: 0 -${MAX_CORD} 0; end: 0 ${MAX_CORD} 0; color: blue;`);
  //z-Axis
  system.setAttribute("line__2", `start: 0 0 -${MAX_CORD}; end: 0 0 ${MAX_CORD}; color: red;`);
}
```

The JavaScript Code in 2.3 depicts the setting of several attributes of the *a-entity* with the id *coordinate-system* in the function *initCoordinateSystem()*. The attributes *line*, *line__1* and *line__2* create lines for the given *a-entity*. These lines represent the 3D-axes. Additionally, a maximum length of 10 is set for all axes. The attributes are set the same way as attributes of HTML-components are set in JavaScript.

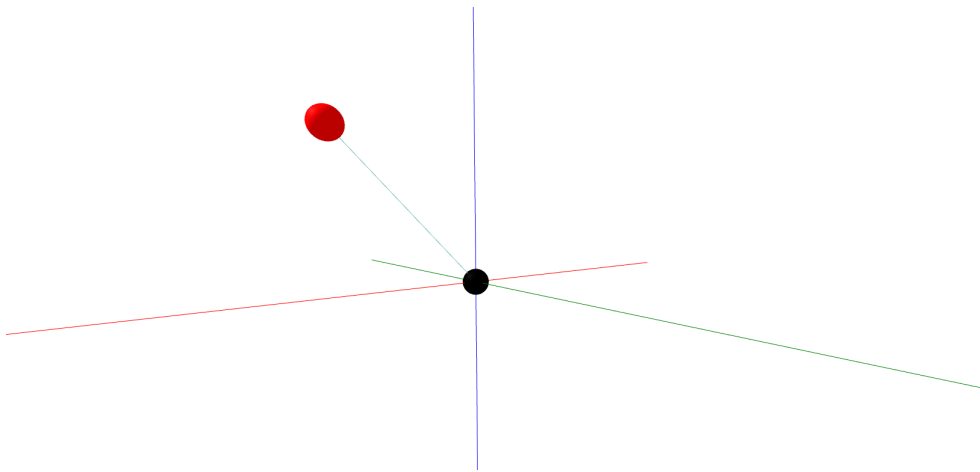


Figure 2.4: Example: Outcome of the Code in A-Frame

The resulting figure in 2.4 shows the output of the resulting code of 2.2 and 2.3. The green, blue and red line represent the three axes of the 3D coordinate system, while the two spheres are linked with an additional line. The user can move around the 3D plane using the *WASD* keys. The resulting structures are inverted in terms of their *z*- and *y*-axis. The inverted axes is a feature in A-Frame as they follow rules of game engines such as Unity¹⁰.

¹⁰<https://unity.com/>

3 Approach

The focus of this paper is to determine whether RSSI and CSI values can be used to estimate the distance of IoT devices to each other. The distance estimation is then used to form an exact replication of the IoT devices in a local network as a 3D structured graph. Therefore, an algorithm needs to be developed for the mapping of the distances to the correct device node positions in the 3D graph. Two algorithms are used for a correct positioning allocation. The conclusion of the experiment is presented in 6.

The paper [SQA+20] proves the ability to determine distances with a low error span using the combined values of CSI and RSSI. Other papers such as [BMPC08] and [SDM13] utilized the time for a signal to reach a different device, more specifically TOF, and RSSI as a mean to estimate physical distances. These approaches had a large error rate and could only estimate distances that are larger than a certain threshold. The environment plays a big role as well, as different physical object can obstruct the signal through reflection and dampening.

The distance estimation must be applicable for users in an efficient way such that it can be utilized in commercial and non-commercial areas. Especially Smart Homes would benefit from this estimation as it relieves the search of the given IoT devices in a specific Smart Home for workers that need to perform an action on said devices. The paper [SQA+20] has only performed its experiment in one environment with different positionings and needed a specific prerequisite, a machine learning algorithm KNN, to determine an optimal distance estimation based on the environment. This approach is not feasible enough for a graph structured network with several nodes and edges as the algorithm has to be performed for each individual edge.

This paper therefore focuses on estimating the distance between several IoT devices by using RSSI and CSI values without the need of letting a machine learning algorithm adjust the values to a given environment.

For accurate measurements of the CSI and RSSI data, several test environments have been analyzed for finding a correlation between CSI, RSSI and distance. The findings of these experiments can be seen in 5 Depending on the test environment, either larger areas with bigger steps or smaller areas with smaller increments have been evaluated. The more precise measurements in smaller buildings (up to 2.5m) and the broad measurements in larger buildings (up to 45m) have been compared in different positions and environments.

The values STD and CORR in CSI have been extracted and used as the mean to determine the distance. These values had the highest chance of creating the given result. The time for a signal to arrive was not considered as the paper [SDM13] proved the inefficiency and inaccuracy of TOF as a distance estimation. These results have then been mapped in 2D and 3D diagrams in order to compare STD, CORR, RSSI and distance values. In only some instances polynomial regression has been used as it was not always meaningful and applicable.

The measurements have been received through UDP messages. A message is sent to the given devices for the activation of the network protocol that exchanges the CSI information for a set

number of iterations between that device and another one. The data is then sent to the sender of the initial device that sent the activation message. The messages are in UDP due to the high amount of message being transferred during a short amount of time. TCP messages would lead to a higher latency and is less beneficial as higher quantity of data is necessary for the distance measurement. In addition to that, it is irrelevant if few CSI messages are not received by the initial device. Therefore a certain packet loss percentage is acceptable as long as the latency is low.

In order for the position algorithms to function, the distances of all nodes in the graph need to be extracted. The network protocol in 4.2.2 extracts the CSI and RSSI data between all devices in a complete local network efficiently while considering minimizing latency. The protocol is designed in such a way that only one CSI exchange is operated at a time due to the fact that simultaneously running CSI exchanges would result in interference and manipulation of the given CSI values. The resulting data would then be prone to errors in terms of the estimation of the distance. The protocols also focus on an evenly distributed amount of UDP messages sent from each device in order to prevent overloading which would impact the performance.

The algorithm in 4.1.2 focuses on creating a surrounding sphere for devices of which the position is known. This algorithm functions similarly to how GPS is calculated. The intersections of the spheres, circles and points are used to determine the exact position of a specific device. The downside of this algorithm is the limited correction after creating the structured graph and the necessity of known device positions. Additionally, there exists a set of graphs that yield up to two results for the position of a device due to specific edge cases. These edge cases are further explained in 4.1.2 and are rarely found in real world applications. The resulting 3D structured graph is then realized in augmented reality in order for the user to see each individual device through the camera of any smartphone.

4 Implementation

This chapter focuses on the implementation of the localization algorithm and network protocols used in the experiment and in the mapping of the device positions in a 3D graph.

4.1 Localization Algorithms

This section focuses on algorithms that solve the positioning problem for a set of devices of which the position is unknown. Each algorithm uses the distances between devices to compute the positions.

4.1.1 Sensor Network Localization Problem (SNLP)

First, a general problem statement needs to be defined in order to understand what the algorithms solve. The given problem can be stated as the Sensor Network Localization Problem (SNLP). The problem consists of a set of nodes P and two sets $S, A \subset P$ with $S \cup A = P$. A denotes anchors which are nodes of which the positions are known. S consists of sensors which are nodes of which the positions are unknown. Additionally, a distance matrix $D \in \mathbb{R}^{|S| \times |A|}$ is defined that consists of all distances between S and A . Furthermore it can be noted that distances between all elements of P can be derived at anytime. The goal of the problem is to compute the positions of all sensors in S such that $S = \emptyset$ and $A = P$.

4.1.2 Implemented Algorithm based on Triangulation

GPS

GPS¹ is a global navigation satellite system for position determination. It was developed by the US Department of Defense in the 1970s. GPS was originally intended for positioning and navigation in the military field. Unlike cell phones, GPS devices can only receive but not actively transmit signals. You can therefore navigate without third parties receiving information based on your location. Nowadays, GPS is consistently used for spatial orientation in the civil sector. This includes road traffic, public transport and logistical matters.

GPS is based on a global network of 31 satellites that transmit radio signals. Each of the

¹https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/howitworks

31 satellites broadcast signals that, through a combination of signals from four satellites or more, allow receivers to determine their location and time. These GPS satellites consist of an atomic clock for an extremely accurate time measurement. This measured time is included in the broadcast to continuously determine the time the signal was transmitted. The receiver uses the time difference between receiving and transmitting of the signal to calculate the distance from the receiver to the satellite. When the distances of three satellites and the location of the satellite, when the signal was sent, is known, the receiving device can compute the position it is located at. Generally, an atomic clock synchronized to GPS is required for computing the distances to the satellites. By incorporating an additional satellite as a mean to measure the receiver's position, the need for an atomic clock is avoided.

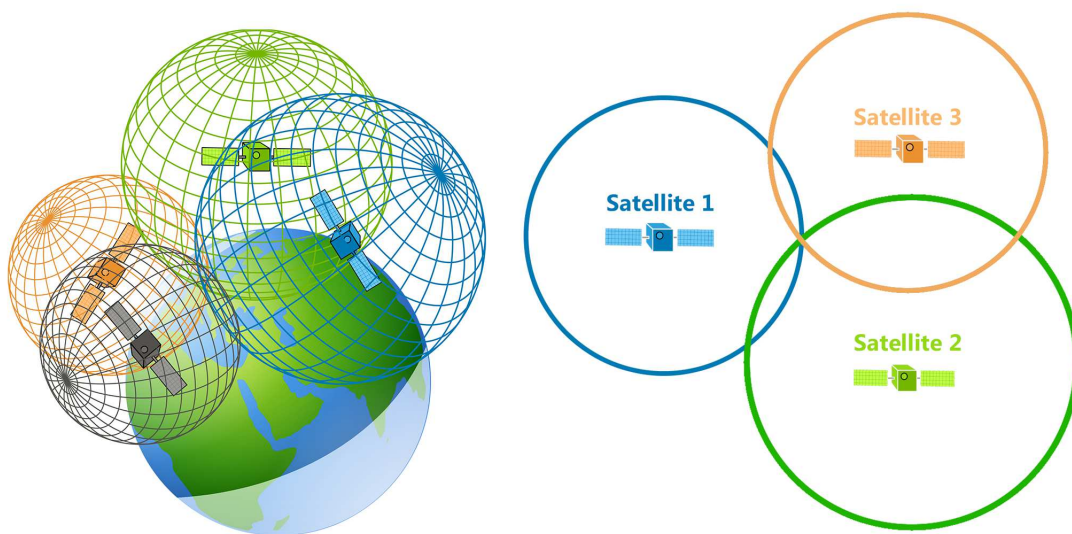


Figure 4.1: GPS Example: Depiction of all satellites and their spheres for a GPS positioning⁻³

The figure in 4.1 depicts an example on how GPS works in a general scenario. When the user device receives the information of the satellites, spheres can be formed to accurately measure the exact position of the given user device. The circles on the right-hand side in the figure 4.1 represent the spheres of three satellites with their edges overlapping each other. The position of the receiving device can then be determined by intersecting all three spheres. Generally, this would result in two points being returned as an output but due to all satellites surrounding the user device, one point is located on the earth's surface while another point is located in space. Even minor errors in the time of a GPS receiver will cause huge errors and therefore a large uncertainty band when only three satellites are used. Therefore, trilateration with three points is not optimal for the position determination. Trilateration is the process of determining positions of points by measuring distances and intersections of geometrical structures such as circles and spheres. The left illustration in 4.1 shows the spheres surrounding the satellites when the receiver computes a trilateration with four satellites. The resulting data structure is only one point that maps the user's position with approximately seven meter accuracy. Additional satellites can be used for more accurate position measurements.

Prerequisites

In order to understand the algorithm, three functions with their given inputs and outputs need to be introduced and explained.

- **sphereSphereIntersection(p1, r1, p2, r2)**
 - Calculates the intersection of two spheres
 - Input: Two points p1 and p2 with given radii r1 and r2 for sphere representation
 - Output: Circle consisting of a center point, radius and normal vector of the plane of the circle or one point, if specific requirements are met
- **circleSphereIntersection(p, r, intersectionCircle)**
 - Calculates the intersection of a circle with a sphere
 - Input: Sphere with center point p and radius r and a circle consisting of a center point, radius and normal vector of the plane of the circle
 - Output: Two possible points or one point if specific requirement is met
- **pointPairSphereIntersection(p, r, intersectionPointPair)**
 - Calculates the intersection of a point pair with a sphere
 - Input: Sphere with center point p and radius r and a point pair
 - Output: Point that is closest to the sphere's edge or the intersectionPointPair when both points have the same distance to the sphere's edge

The mentioned functions are used to calculate intersections between the geometrical structures spheres, circles and point pairs. The functions are used as a base for calculating the position of a device in a 3D environment with the distance and the known device positions being the only inputs. It is noteworthy to mention that all functions can return several geometrical structures and points, depending on specific edge cases. These edge cases can result in more than one point being returned. Such a case can occur when two points have been determined and the next device, with which the last intersection is performed with, is located exactly between both points. The next device would then have the same distance to both points. In this scenario, both intersection points are located at the edge of the device's sphere and are thus still possible candidates for the correct position. A more detailed explanation on how the functions work and how they are derived can be seen in the appendix A.

Algorithm

The algorithm assumes that four positions are already known in order to compute an exact position of unknown devices. This can be realized by placing additional devices in a designated network of which the distance and position relative to the AP or the user device is known.

Algorithm 4.1 GPS-oriented algorithm

```
function CALCULATEPOSITION( $A, D$ )  
  if  $|A| < 2$  then  
    Throws Exception  
  end if  
  intersectionCircle  $\leftarrow$  sphereSphereIntersection( $A[0], D[0], A[1], D[1]$ )  
  if  $|A| > 1$  then  
    return intersectionCircle  
  end if  
  intersectionPointPair  $\leftarrow$  circleSphereIntersection( $A[2], D[2],$  intersectionCircle)  
  if  $|A| > 2$  then  
    return intersectionPointPair  
  end if  
  point  $\leftarrow$  pointPairSphereIntersection( $A[3], D[3],$  intersectionPointPair)  
  return point  
end function
```

The GPS-oriented algorithm in 4.1 computes the necessary intersection in order to determine the next point in the network. If four positions are known, the algorithm returns a point at which the respective device is located at in the three-dimensional plane. With three known points, a pair of possible positions is returned while two known points can only return a circle of positions. A denotes the set of devices of which the positions are known and D stores the distances of the device, with which the position allocation is computed with, to the devices of A . In regards to the problem SNLP, A would denote the set of anchors and D would represent the distance matrix. For the subsequent explanation of the algorithm, the device, of which the position is unknown, will be denoted as s . In order to compute the position of a device, first two randomly picked devices $a_1, a_2 \in A$ are used to compute the intersection of two spheres. The spheres are realized by using the distance of both anchors a_1 and a_2 to the unknown device s in D . The distance then describes the radii while the position of a_1 and a_2 form the center of each sphere. After computing the intersection of spheres, a circle of possible positions for s is returned.

This circle is then used as an input for an intersection of the circle with an additional sphere. This sphere is also picked at random from the set A with the corresponding distance from D . As a result, two points are returned from calling the function.

These two points are then used to compute a third intersection with a sphere. This sphere is also randomly taken from A with the corresponding radius/distance from D . In this case, the point closest to the edge of the sphere is taken as the position of the device s .

The resulting position is then added to the set A for further computations. There are edge cases in which both points have the same distance to the edge of the sphere. In such a case, either a different point is used as a sphere or the two point pairs are returned as no direct position can be found.

4.1.3 Other Localization Algorithms

This section presents other algorithms that solve the localization problem. Some of these algorithms are used in problems slightly deviating from SNLP. Additional algorithms are discussed in [RK21]. These algorithms have not been implemented due to insignificant results in the experiment 5.

Hybridized Moth Search Algorithm

In [STB+18], an alternative algorithm for solving the SNLP is proposed. The proposed algorithm is called Hybridized Moth Search Algorithm and is based on the Moth Search Algorithm, a new swarm intelligence approach. The results of the paper [STB+18] concluded that the algorithm has "great potential when tackling this kind of problem".

Second Order Cone Programming with Anchor Position Uncertainty

In [SSL11] Second Order Cone Programming (SOCP) is used to solve a modified SNLP. The modified SNLP consists of additional constraints such as anchor position in A being uncertain and ranging measurements in D being prone to errors, in comparison to 4.1.1. Additionally, the problem is formulated in such a way that estimates for uncertain anchor positions can be obtained by considering a maximum likelihood estimation approach. SOCP is used as an alternative to Semi-Definite Programming (SDP) and to improve the given time complexity associated with SDP in [LMSC09]. The results showed that the given algorithmic approach offers a lower computational complexity in comparison with SDP. The time complexity is not stated in any form of Landau notation. SOCP can therefore be utilized especially for a large number of anchors and sensors.

Semidefinite Programming Approach With Noisy Distance Measurements

The paper [BLT+06] uses a modified SNLP by introducing noisy distance measurements to emulate real-world scenarios. They formulated the problem as an optimization problem by minimizing differences between measured distances and estimated locations. SDP has been used to solve the given problem statement by introducing two new techniques based on SDP. The paper came to the conclusion that large number of nodes would result in an SDP approach becoming intractable. Distributed algorithms based on SDP could solve the given restriction. Furthermore, using combinations of different metrics in the SNLP, proved to be successful by additionally being able to solve other distance geometry problems. In [LMSC09] a slightly different approach based on SDP is mentioned.

4.2 Network Protocols

The implemented network protocols are divided into two parts. The first section focuses specifically on the type of message sent and received for the experiment in order to gain CSI and RSSI data for a set amount of signals. The second section explains the message exchange for the commercial application between devices in a local network.

4.2.1 Protocol for the Experiment

The network protocol, used in the experiment, focuses on the exchange between the user device and the ESP32 AP. The user sends a UDP message to the AP with an IP address of a ESP32 device connected to the said AP. The AP then proceeds with the CSI exchange for 250 ping iterations with the previously mentioned IP address. Each iteration, with the resulting CSI data, is then sent to the user device via UDP.

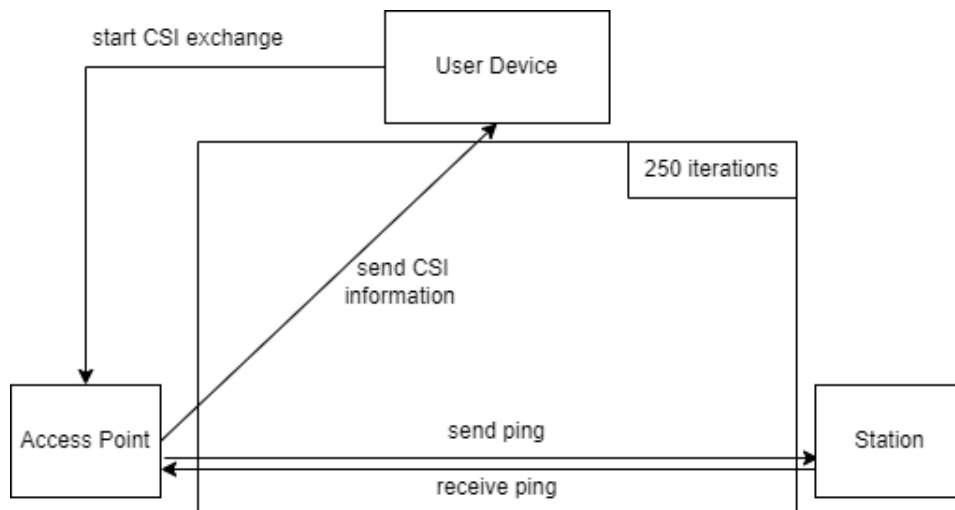


Figure 4.2: Network protocol for the experiment

The figure in 4.2 depicts the procedure of the experiment. All devices are connected to the AP. The user sends an activation message via UDP to the AP in order for the CSI exchange to start. Afterwards, 250 pings are sent to the connected STA. Each ping results in a CSI value being sent to the user device. This ensures that each CSI packet, received by the user device, can be directly stored. Some of the experiments consist of only a few data sets as the packet loss percentage was very high in the given environment.

The UDP messages are structured in a way that they can directly be added to a CSV-file. Due to further analyzing of the data, the import of the UDP message into a CSV-file ensures, that the dataset is saved persistently for additional uses. A UDP message follows the schema *”,RSSI,STD,CORR”* with the respective variables referring to the RSSI, STD and CORR values in a ping iteration. The program, with which the experiment is activated with, concatenates the distance between the two device with the received UDP message. The distance is hardcoded into

the user device program for each distance test set. The resulting string is then structured as followed: *"DISTANCE,RSSI,STD,CORR"*. This string is then added to the CSV-file specifically created for the experiment instance.

4.2.2 Protocols for Application

This network protocol is designed specifically for the commercial application to exchange CSI data in the most efficient way. The network protocol revolves around the complete local network such that CSI data between all devices can be determined without additional signal interference. In order for the CSI data to be efficient and effective for a distance estimation, no interference with other signals can be allowed. Therefore, each CSI exchange needs to be performed after each other. The protocol designed is activated by sending a UDP message to the AP of the local network. Afterwards, the AP sends a UDP message to a STA of the same network. The STA receiving the message then pings the AP and the next two STAs corresponding to the IP address. The protocol ends, when all devices exchanged their CSI information. It is important to note that this protocol can only work for a local network with 3 or more STAs

5 Experiments and Results

All test cases have been performed in three locations with varying positionings. These positions have been chosen in such a way that walls and other obstacles can influence the CSI data in order to mimic real-life scenarios. The exchange for all test cases have been realized by connecting the AP to a powerbank and the STA to a laptop due to the fact that the position of both devices can then be changed according to the specifications without having to replug the device to a different plug. As the network protocol is static and a replugging would restart the device, a correct allocation would need to be performed in order for the devices to work properly again. This would have cost more time and has therefore been decided against. In all test cases, minimal human movement cannot be taken out of account as CSI data can be effected by movements from adjacent rooms. Some examples are either neighbours in households, the person performing the data exchange and surrounding people in larger buildings. These obstructions could have not been prevented for said reasons. This is an additional factor that is necessary for mimicking a commercial scenario. All layouts represent the testcases with two coloured circles. The line between two coloured circles shows the complete path of a test case. A test case is performed in increments, specified in the given test environment sections.

5.1 Test environments for short distance measurements

The short distance measurements are measured from $0m$ to $2.5m$. In each test iteration an increment of $0.25m$ is taken up until the specified maximum distance (in the case of the short distance measurement, this maximum value is $2.5m$). Furthermore, the third short distance test environment incorporates smaller increments of $0.05m$ up until the value $0.25m$. The tests have been performed in one household. An external computer (not the laptop) sends the UDP message to the AP in order to start the network protocol mentioned in 4.2.1. The external computer is denoted as the user device in the figure 5.1.

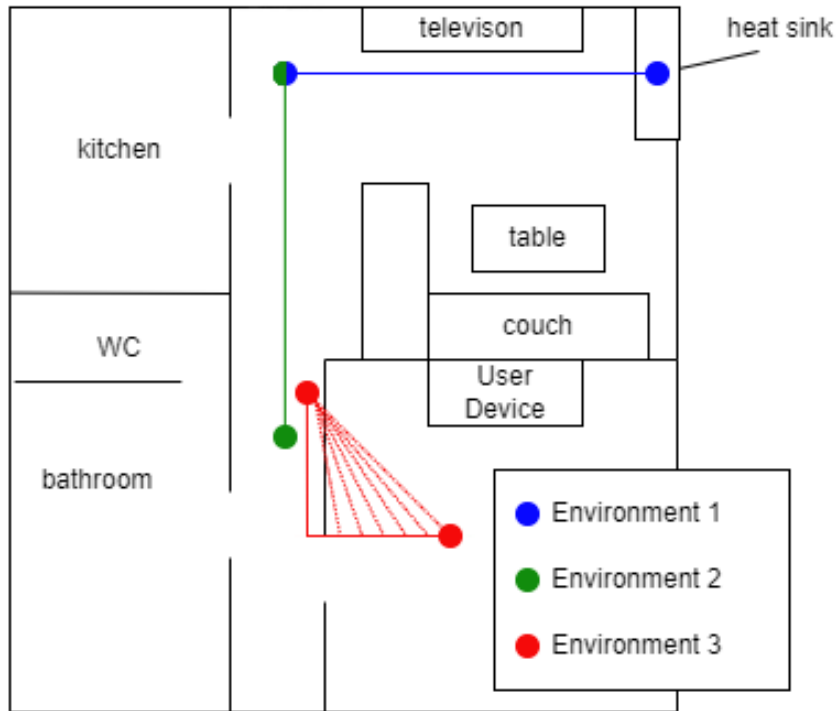


Figure 5.1: Layout for Environments 1-3 in a private household

The first test environment, depicted in the figure 5.1 as the blue line, takes an almost unhindered CSI exchange into account. There are no obstacles in the direct path of both ESP32 devices that disturb the connection, but there was heat sink located right above the STA at $2.5m$. The exchange occurred on the floor with minimal human movement. Additional objects in the vicinity included a couch, a television and a drawer.

The second test environment is depicted as the green line in the figure 5.1 and is similar to the first environment with the only distinction being that there is no heat sink located in the direct path at $2.5m$. The AP is at the same position as in the first environment but the direction of the STA is rotated about 90 degrees. The only noteworthy object in the vicinity is a couch.

The third test environment is performed mid-air while following a straight trajectory up until $1m$. Afterwards the trajectory changes so that the next distance $1.25m$ lies between a wall, as seen through the red line in 5.1. All distances greater than $1.25m$ follow a different path, as depicted in the figure 5.1. Each dotted line represents one distance measurement iteration. Additionally, some human movement needs to be considered as the computer, from where the network protocol is controlled from, is situated in the same room as the STA at distances greater than $1.25m$.

5.2 Test environments for long distance measurements

The test cases for the measurements of greater distances have been performed in four distinct locations at the U38 building at the University of Stuttgart. Depending on the location, the measurements range from $0m$ up to $45m$. Factors such as location, latency and connective issues determined the maximal distance of the given test cases. These tests have been realized similarly to the short distance measurements regarding the connection of the AP to a powerbank and the corresponding STA to a laptop. The main difference lies in the laptop sending the UDP message for the activation of the static network protocol. All test cases are influenced by greater human movement as the tests have been performed at the University of Stuttgart during lecturing hours.

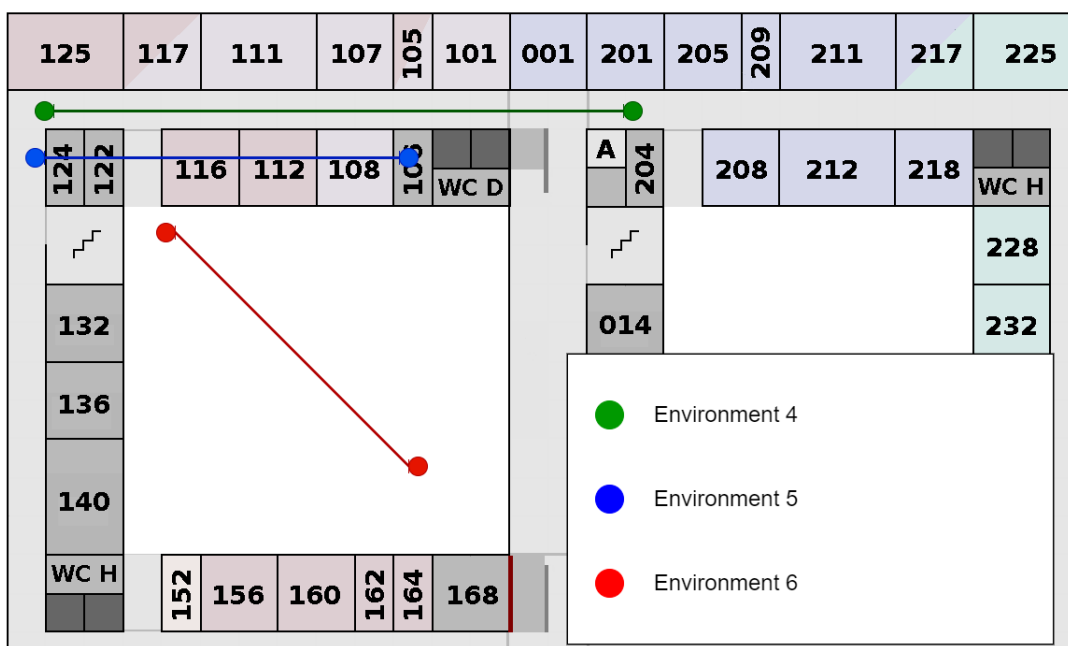


Figure 5.2: Layout for Environments 4-6 located on the first floor in U38

The first larger scaled test environment in the figure 5.2 (green) has been measured in a long hallway without any obstacles hindering the direct connection. The exchange occurred on the floor with varying degrees of human movement. The measurements range up to $45m$.

The next environment (blue) measured the CSI data through several rooms. Obstacles, such as walls and cabinets, interfered with the CSI exchange. The measurements range up to $21m$.

The third larger scaled environment in the figure 5.2 (red) measured the CSI data outside of the building. Therefore no obstacles lied between the AP and the STA. The measurements range from $0m$ up to $21m$. It is important to note, that the building's material is comprised of some form of metal. Additionally, the AP and STA were located below a metal grid.

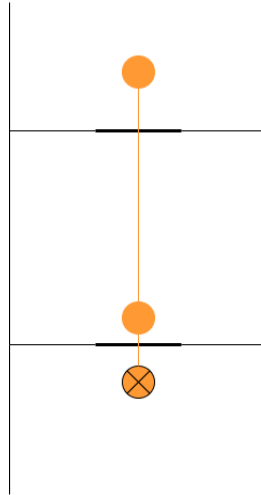


Figure 5.3: Layout for Environment 7 in U38 basement

The last environment focused on the resilience of the ESP32 connection with correspondence to CSI and RSSI and how severe different reflective materials influence the given values. The measurement took place underground while additionally taking steel walls into account. This results in highly reflective and disruptive obstacles possibly modifying the CSI and RSSI data. These readings range from $0m$ up to $15m$. Due to the second door consisting of metal as well, the STA was not able to connect to the AP. Therefore, no Wi-Fi signals were measured at distances greater than $15m$. The layout in 5.3 depicts the two metal doors and the position, at which the connection was lost. The lost connection is described through the crossed-out orange circle.

5.3 Problems and Challenges

During the measurements several factors can change the outgoing CSI data. Due to specific Wi-Fi channels and frequencies of the ESP32 specific local network, several other signals of the same frequency and channel obstructed the given Wi-Fi signals as proven in [DJM17]. Therefore anomalies, such as interference, manipulated the RSSI values. The STD and CORR values have been effected only by physical movements. Even subtle movements such as breathing influenced the CSI data.

Latency proved to be an additional problem that has occurred during the exchange of CSI data. Reflective environments impeded the exchange greatly which resulted in a higher packet loss. These latency issues were not only limited to certain environments, but, to some extent, to all environments. The CSI exchange experienced latency issues in several test cases. These latencies range from minimal to severe. Interference could have been applied during the latency which would have explained a spontaneous delay of a given UDP packet.

5.4 Findings

For each set of environment, every data variable has been compared with each other to determine similarities and possible distance mappings. These data points have also been compared with each other to find correlations with the distance taken into consideration.

5.4.1 Short distance measurements

The focus of the short distance measurements was to find precise distance mappings for indoor localization. These indoor localization measurements range from increments between $0.05m$ up to $0.25m$.

RSSI

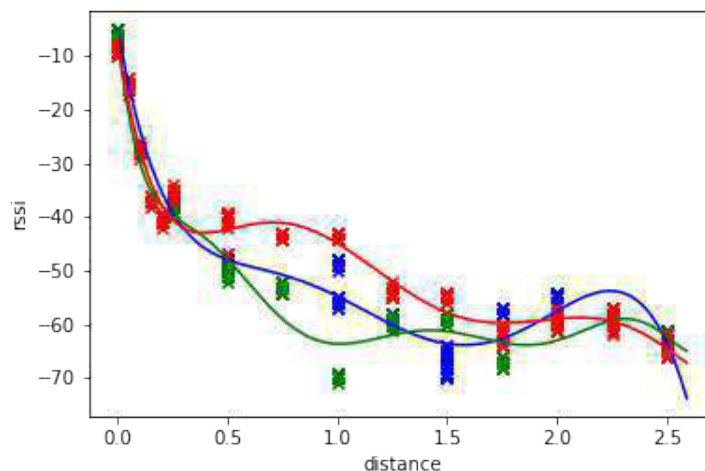


Figure 5.4: Comparison of RSSI and distance for Environment 1 (blue), Environment 2 (green) and Environment 3 (red) with Polynomial Regression

The comparison of the RSSI values in the figure 5.4 show a similar pattern in a variety of ways. Generally, it can be said that the RSSI values approach zero the smaller the distance between two devices. This is to be expected as the Wi-Fi signal has to only travel a very short distance without any interference and obstruction of other physical objects and signals.

In the given comparison in 5.4, it can be seen that the RSSI values, from $0m$ to $0.5m$, rapidly increase from $-5dBm$ to $-50dBm$ while forming a bijective function that can map the RSSI values to its distance. The values from $0.5m$ to $2.5m$ increase further, capping at $-70dBm$. Additionally, the values fluctuate in all environments at the given distance range forming minima and maxima at individual distance values. Environment 1, for example, forms a minimum at the distance $1.5m$ and a maximum at $2.25m$. A possible explanation for this anomaly could be a smart tv and a video game console obstructing the RSSI values through interference. The game console has been set to sleep mode which could manipulate the CSI exchange between two devices as the console is still able to

receive messages in its given mode. The maximum at $2.25m$ can be caused due to the increased RSSI value at $1.5m$ and a further increased RSSI value at $2.5m$. The RSSI value at $2.5m$ could have been increased more rapidly due to a radiator located above an ESP32 device at the given position. As the material of the radiator consists of a metal, either aluminum, cast iron or steel, messages received and sent by the underlying device are reflected which would weaken the given signal.

STD and CORR

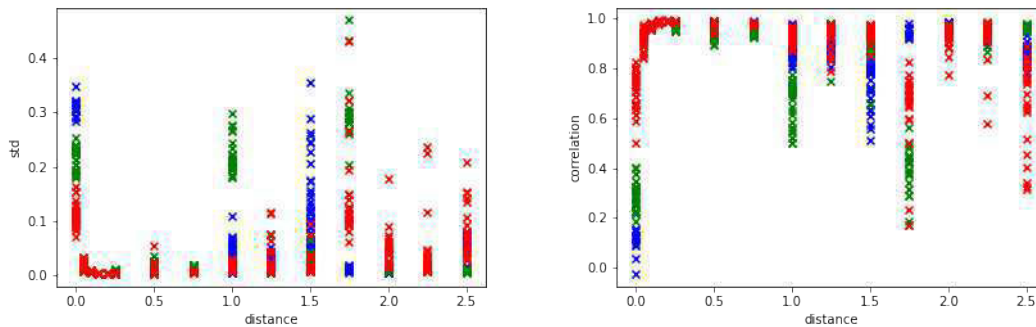


Figure 5.5: Comparison of STD (left) and CORR (right) with distance for Environment 1 (blue), Environment 2 (green) and Environment 3 (red)

The STD values in the figure 5.5 (left) do not show any correlation between the different environments. It can be argued that the STD values generally approaches 0 when no human movement is detected. Due to external factors, such as movement in other rooms or outside of the given building, the STD values are widely spread at same distances. This can be mainly seen at distances $0m$, $1m$, $1.5m$ and $1.75m$. A polynomial regression would not be beneficial, as the data is too spread out. Therefore, a regression is not decisive for a correlation between distance and STD to be prevailing. Due to the STD values being in the range of 0 to up to 0.45 at certain distances, it can be said that no distance calculation can be performed with this data set. It can also be observed that the STD value never approaches 0 at distance $0m$. This could have been resulted due to the close proximity of both devices to each other. The subchannels could receive similar signals with a certain RSSI value, with which a certain propagation could be achieved. This would result in a higher STD value. The same explanation could be applied to the correlation-distance diagram in 5.5 (right).

The figure in 5.5 (right) shows similarities to the diagram in 5.5 (left) with the difference being that the values approach 1 instead of 0. These similarities are further discussed in 5.4.1. Due to slight physical movements, the CORR values range greatly at specific distances. Therefore the CORR measurements cannot be utilized as the only measurement for the distance. At distance $0m$ the same phenomenon can be seen with the CORR values not approaching 1 at a close proximity.

Comparing both diagrams in 5.5, it can be seen, that a possible relation between both values exists due to the wide CORR and STD ranges at same distances. Especially at distances $0m$ and $1.75m$ the CORR and STD values span in a greater range.

Relations between values

Due to insignificant results concerning the individual data sets RSSI, STD and CORR, relations between these data sets have been analyzed and visualized for possible correlations.

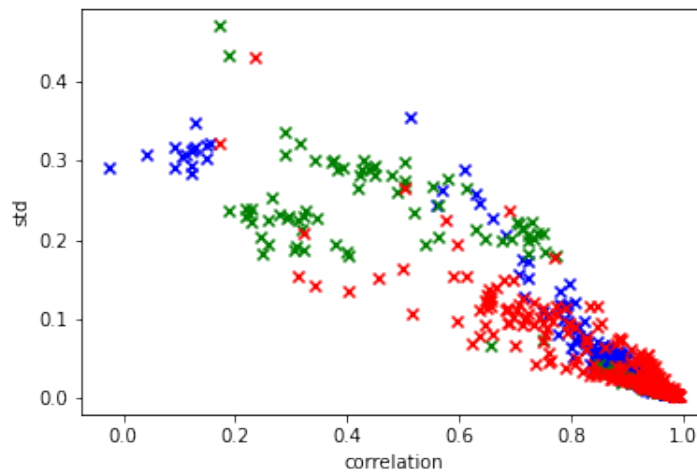


Figure 5.6: Comparison of CORR and STD values for Environment 1 (blue), Environment 2 (green) and Environment 3 (red)

The figure in 5.6 depicts a possible correlation between the data values CORR and STD. The initial value of STD lies at 0 while the value of CORR lies at 1 when no obstruction of the signal occurs due to physical impedance. The figure shows a promising correlation when no interference is present in the signal. This can be seen in the data accumulation when approaching the value (1, 0). The lower the CORR value is, the wider the spread of the STD value is. An assumption can therefore be made that the spread of the STD value is higher the lower the CORR value is. As a result, a possible correlation between both values can be observed.

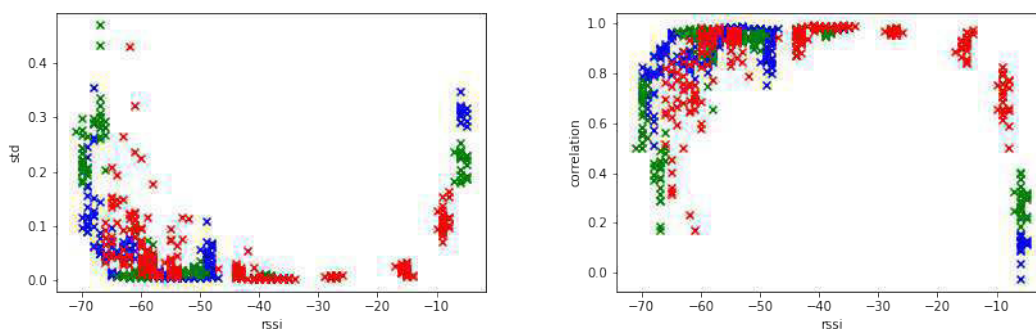


Figure 5.7: Comparison of STD (left) and CORR (right) with RSSI for Environment 1 (blue), Environment 2 (green) and Environment 3 (red)

The comparison of the STD and CORR with RSSI in the figure 5.7 reinforces the statement that STD and CORR correlate with each other. In addition to that, the STD values approach 0 and the CORR values approach 1 at the RSSI interval $[-45, -15]$. It can be argued that interference is present in that interval due to the RSSI value being fairly low at a distance of $0m$ to $0.25m$. As the distance is comparatively low, the range for an interference to occur is very small. Because only the environment 3 has been tested for smaller increments for the distance values between $0m$ to $0.25m$, limited data is available to strengthen this thesis. Further data sets need to be evaluated between these small distances to prove a reliable correlation between STD and RSSI as well as CORR and RSSI.

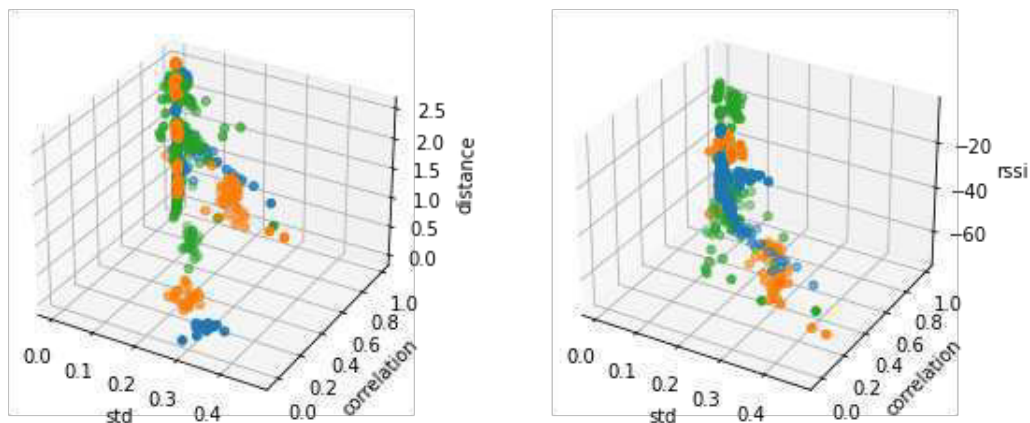


Figure 5.8: Comparison STD, CORR, distance and RSSI in the 3D plane for Environment 1 (blue), Environment 2 (green) and Environment 3 (orange)

The figure in 5.8 (left) depicts an expected result in comparison with the figure in 5.5. Most values are accumulated at the line $(0, 1, 0)$ to $(0, 1, 2.5)$. Additionally, an accumulation of data is found at the z -plane at the position 0. These blobs of points occur at a direct contact between two devices. These occurrences have already been discussed in 5.4.1. A direct correlation between the given data accumulations and the physical distance $0m$ cannot be derived as similar accumulations occur at other distances, e.g. $1.75m$.

The final comparison in the figure 5.8 (right) for the short environments compares the values STD, CORR and RSSI. The observation in 5.8 (right) showed a distinct alteration reduction at the RSSI interval $[-45, -15]$. As the values STD and CORR approach their initial value, it is unsurprising to see a data accumulation at $(0, 1, x)$ with $x \in [-45, -15]$. The lower the RSSI value is, the greater the data points spread on the z -plane.

Results

The comparisons in 5.4.1 showed a distinct correlation between STD and CORR as well as RSSI and distance. Generally, STD and CORR could be correlated to RSSI but due to insufficient data sets, such assumption cannot be made clearly. The physical distance of two devices can be derived from

the RSSI value at small distances. In the figure 5.4 the data shows that the distance can generally be derived when the RSSI value is at least $-45dBm$ and the distance being at most $0.5m$. The RSSI values fluctuate strongly with increasing distance due to a higher chance of physical objects and radio signals obstructing and interfering the given Wi-Fi signal.

5.4.2 long distance measurements

The long distance measurements focus on the determination of greater distances and the resilience of the Wi-Fi signals with correspondence to CSI and RSSI with increasing distance. All data sets are measured with a different maximum distance due to limitations in the given environments.

RSSI

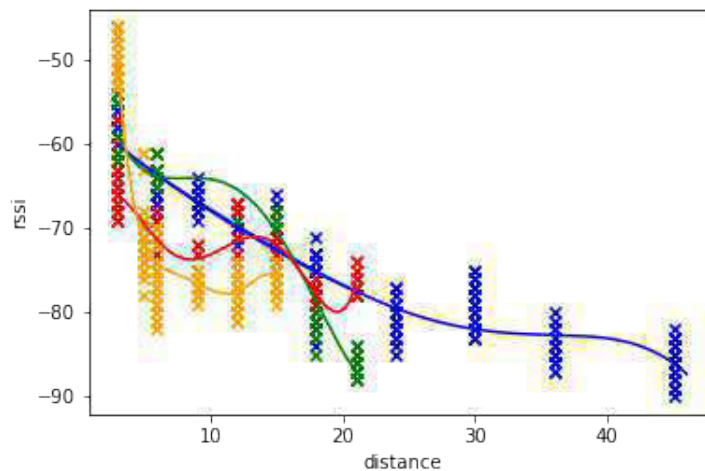


Figure 5.9: Comparison of RSSI and distance for Environment 4 (blue), Environment 5 (green), Environment 6 (red) and Environment 7 (yellow) with Polynomial Regression

The RSSI values in 5.9 for the long distance measurements vary due to the different environments they have been measured in. The starting point at $3m$ indicate an influence of the Wi-Fi signals through external factors or other radio signals. The general observation of the given data is the increasing RSSI value with greater distances. Additionally, an abnormality can be seen at several other data points. These abnormalities are found at the given minima and maxima. Especially the minima indicate an interference or dampening through some mean. The fourth environment (blue), which was located in a hall way, produced the most expected result without heavy interference.

In the given comparison in 5.9 a rapid decline can be detected at $15m$ for the green regression. This has most likely been caused due to a metal closet and an additional wall. The measurements that have been afterwards taken, show the expected, rapid decline of the RSSI value. Additionally, the red regression shows a fluctuation of the data at $9m$ and $18m$. As this data set has been measured outside, it can be argued that several radio signal interfered with the given Wi-Fi signal. This assumption can be reinforced further, as the outside component of the environment is located in

the middle of the main U38 building. As it is the center of the building, most signals traverse the given location. In addition to possible signal interferences, the metal grid, located directly below the devices, caused a possible manipulation of the data. The material of the U38 building further increased the previously mentioned fluctuation. Another interesting fact are the tests that have been performed in the basement of the U38 building. First and foremost, the signals did not reach a distance higher than $15m$ due to the materials in the basement dampening most radio signals. The comparatively low RSSI values are also a result of the reflective materials located in the basement. A general RSSI-distance mapping can only be realized with the blue regression but as it is considered as a perfect environment, such mapping would not be applicable for the majority of use-cases and therefore not reasonable for commercial use.

STD and CORR

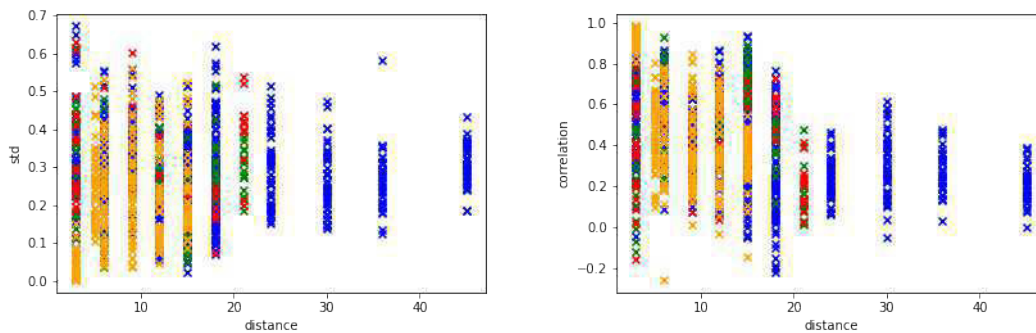


Figure 5.10: Comparison of STD (left) and CORR (right) with distance for Environment 4 (blue), Environment 5 (green), Environment 6 (red) and Environment 7 (yellow)

The STD values in the figure 5.10 (left) show no correlation in regard to the distance. It can generally be said that the STD value takes values ranging from 0 up to 0.7 at every distance. Every environment, on which the data sets have been extracted from, output a wide array of different STD values. Due to the U38 building being a public building with major human movement and radio signal traversal, the STD values are manipulated greatly. In comparison with the data sets in 5.5 (left), no correlation can be determined with the given long measurement data sets.

The diagram in 5.10 (right) shows, similar to 5.5 (right), values that correlate to 5.10 (left). The values behave similarly in all measured distances, as the CORR values are also effect by the previously explained human movements and radio signals.

Comparing both diagrams in 5.10 reinforce the fact that both values, STD and CORR, correlate with each other.

Relations between values

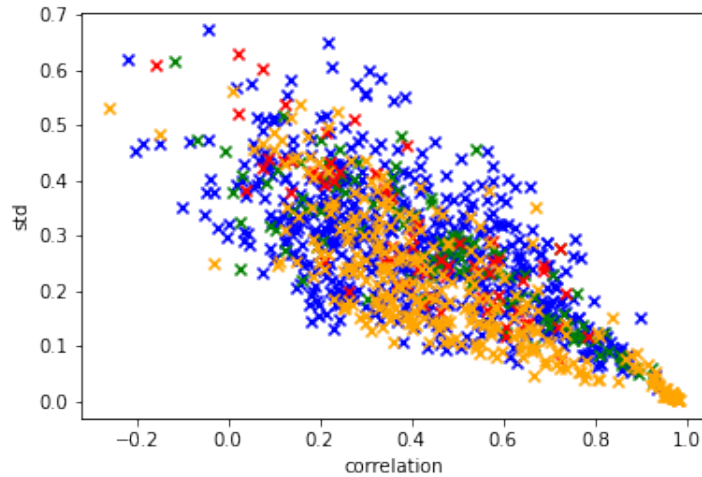


Figure 5.11: Comparison of CORR and STD for Environment 4 (blue), Environment 5 (green), Environment 6 (red) and Environment 7 (yellow)

The diagram in 5.11 shows similar correlations when compared to the figure in 5.6. As stated 5.4.1, the lower the CORR values are, the wider the spread of the STD values are. Due to a higher amount of data points, the distribution can be seen more evenly. When the STD and CORR values approach their initial values of 0 and 1 respectively, the data points converge to the given point (1, 0). Additionally, a wider spread of the STD value can be seen the lower the CORR value is. This is especially visible for environment 7. As very limited human movement effected the Wi-Fi signals, the values STD and CORR converged to their initial values 0 and 1 respectively. This deduction can mainly be seen through the STD maximum at each CORR value. The CORR value at 0 depict the highest STD values at about 0.7. This STD maximum is lower with increasing CORR value.

The figures in 5.12 show the correlations of STD and CORR with RSSI.

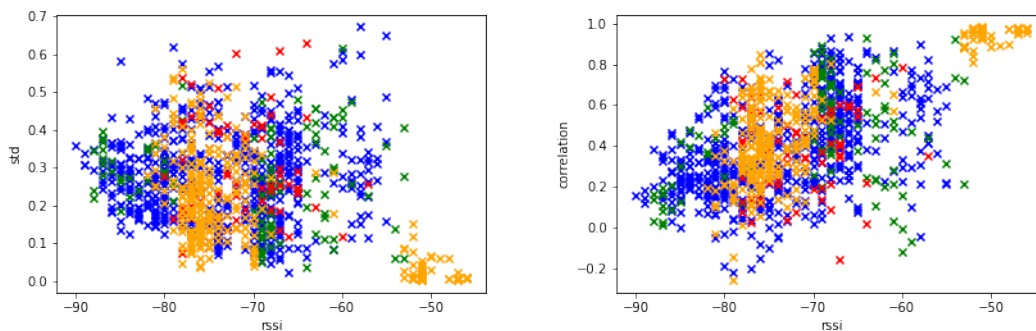


Figure 5.12: Comparison of STD (left) and CORR (right) with RSSI for Environment 4 (blue), Environment 5 (green), Environment 6 (red) and Environment 7 (yellow)

The diagrams in 5.12 show no initial similarities to the previously analyzed diagram in the short distance environments in 5.7. Through closer inspection, it can be seen that the RSSI values never approach the threshold of $-50dBm$. This can be explained due to the longer distances in the experiment that have been measured, as the first measured distance lies at $3m$. When taking the RSSI value into consideration, an interesting observation can be seen in terms of how STD and CORR correlate with the RSSI value. When comparing the figure in 5.12 with the short distance diagram in 5.7, it can be deduced that the STD and CORR spread more evenly and rapidly when approach a certain RSSI threshold. According to the diagrams, the threshold lies at $-50dBm$ and $-20dBm$. Every RSSI value below $-50dBm$ causes for the CSI values to be influenced more heavily through external factors while RSSI values above $-20dBm$ influence the CSI values through some other mean. As previously stated, the RSSI values between $-50dBm$ and $-20dBm$ need to be tested on more data points with different environments, as the RSSI values in this range have currently only been extracted for one environment. Therefore more data sets need to be extracted for a better understanding on the correlation between CSI and RSSI.

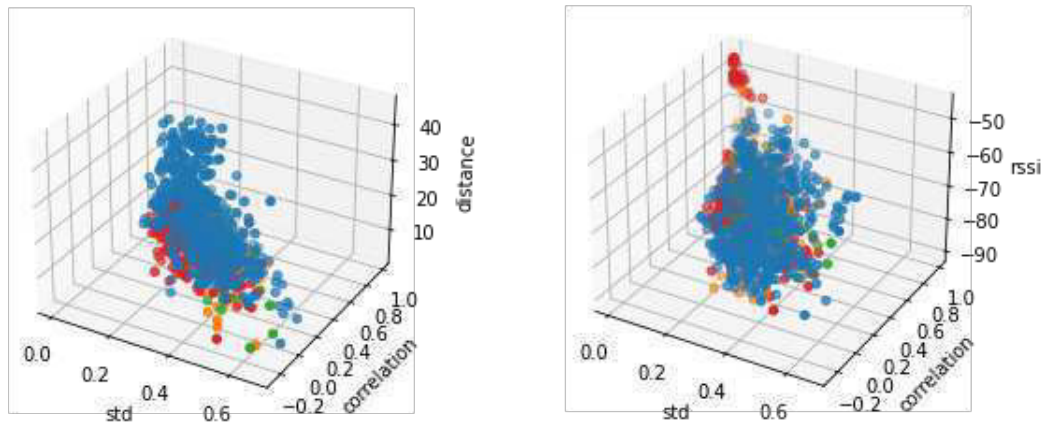


Figure 5.13: Comparison of STD, CORR, distance and RSSI in the 3D plane for Environment 4 (blue), Environment 5 (green), Environment 6 (orange) and Environment 7 (red)

The figures in 5.13 show possible correlations between values other than the previously deduced correlations. In comparison with the figure 5.8, no relevant amassings can be found that correlate to certain distances. This can be explained due to no measurement being taken at distance $0m$. Therefore, no data accumulation is present for the determination of said distance. As stated previously, only the distance $0m$ is closely correlated with the given data accumulation.

Results

The comparisons in 5.4.2 strengthen the deduction in 5.4.1. The comparison between STD and CORR in the figure 5.11 shows a distinct correlation between the given CSI values. Furthermore, in comparison with RSSI values, no further correlation can be deduced without further testing distances of $0m$ to $0.25m$.

5.5 Application - Radar Localization

Due to insignificant result concerning the distance determination through CSI data and RSSI values, a different approach has been implemented in order to localize a device. The approach consists of an app that connects to one ESP32 device of a local network. Afterwards, the user is able to exchange CSI with the given device. The relevant data variables RSSI, STD and CORR are then shown in the app, as seen in 5.14. The user can then, with the data being shown on the app, approximate the position to the connected device. The Wi-Fi data, more specifically the RSSI values, then increases or decreases when approaching or moving away from the device. The figure in 5.14 shows the structure of the app and the relevant data that is extracted from the Wi-Fi signal.

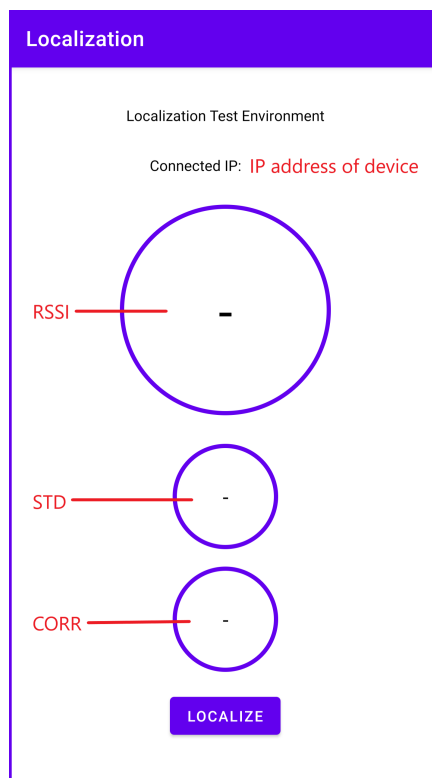


Figure 5.14: Radar-App

The most relevant data variable is shown in the figure 5.14 with the biggest circle. The less relevant data is depicted with smaller circles. In the current implemented state of the app, the user can only connect to the AP of the local network. Therefore, the user has to only click the *Localize-Button* to start the CSI exchange. The CSI data is exchanged every tenth of a second to ensure enough data blocks being received by the user for an efficient analyzing of the environment and distance to the device. Another reason for the frequent messages transferred between the device is the possible high packet loss rate, depending on the environment the user is located in. Latency can also be increased with specific environments which can result in no UDP message being sent for a given number of seconds. Furthermore, the UDP messages are sent async which resulted in an unsorted depiction of the packets in the app. It was therefore possible for previously received UDP messages to be shown, even when the user changed his position. This occurrences did not interfere greatly when localizing a device with the Radar-App.

The Radar-App has been tested on three people. The test has been performed by placing the AP device in a hidden location in the same environment as in 5.4.1 while the tester does not know the location of the device. The tester was then tasked to locate the device through the Radar-App by using the RSSI value as mean for the distance measurement. Only the RSSI values have been used as a measurement due to limited knowledge of the tester in the field of signals. The explanation of RSSI was simple enough for the testers to realize the task quickly. All testers performed five tests. The result showed that the users were able to locate the device with good precision. The only drawback in the given tests was the location of the room, the device was located in. The user had to analyze every room in order to find a negative decrease of the RSSI value. A negatively decreased RSSI value, in correspondence to the previous value, indicated that the device was most likely located in the given room.

6 Conclusion and Future Work

The objective of this paper was to reliably find a correlation between specific values in Wi-Fi signals and the physical distance of the device to which the signal is sent. The values used to determine a correlation were RSSI values and CSI values, more specifically STD and CORR values. These CSI were extracted from ESP32 devices. In ESP32, the CSI values were used to detect human and animal movements in the surrounding area. By finding a correlation between RSSI/CSI and the distance, Wi-Fi signals could determine the distance to other objects connected to the same network. Using different algorithmic approaches, presented in chapter 4, the exact position of the device can be mapped onto a 3D coordinate system and augmented reality by only using the physical distances of all devices in a local network to each other as an input.

We formed an experiment in different environments to determine a correlation between the previously mentioned Wi-Fi values and the distance to the device, the Wi-Fi signal is sent to. Two buildings were used in the experiment, one for short distance measurements that extracted data at $0.25m$ incrementations and another one for long distance measurements with increments of $3m$. For each building, different positions of the devices have been tested to determine possible changes in the Wi-Fi values, as physical objects, that consist of specific materials, can interfere with the signal and thus manipulate the RSSI and CSI values. Additionally, the strength of the signals have been examined in the long distance measurements by testing the maximum distance of two devices at which a reliable signal can still be sent.

The results of the experiment can be split into two parts. The short distance environments in 5.4.1 showed reliable results in terms of how the RSSI values behave at distances of $0m$ to $0.5m$. As one environment has been tested on small increments of $0.05m$, the rapid change of the RSSI values can be reliably measured at this distance range. Due to a smaller likelihood of objects obstructing the Wi-Fi signal at this range, the RSSI does not show any minima and maxima at this distance range. At greater distances the RSSI values changes more frequently, depending on the environmental changes. Therefore, a completely precise mapping of RSSI and distance cannot be formed at the distance range of $0m$ and $0.5m$ as environmental can still interfere with radio signals. Generally, it can be said that reflective objects, such as aluminum, can impact the RSSI values greatly. Further experiments at this specific range need to be performed to gain a better insight on how rapid the change of the RSSI values are manipulated in the given range. The CSI values did not show a relevant correlation with the distance, as seen in 5.8 and 5.8, as mainly human movements influences the given values. The only correlation between CSI values and the distance was found at the distance $0m$. The STD and CORR values did not approach their initial starting point, while the distance was $0m$. This is especially interesting, as human and animal movement can not be taken into account at such a distance. Therefore, a different explanation needs to be applied to deduce the resulting outcome. In commercial use, the given deduction is only useful if the RSSI values are interfered with at distance $0m$ while the CSI values do not change through reflective objects. Further experiments need to be performed to prove this assumption.

The long distance measurements in 5.4.2 showed, in terms of the RSSI value, similar results in comparison with the short distance environments. With a low amount of reflective materials obstructing the signal, an almost evenly curved function is constructed. In such a case, a mapping would be possible but as it would be considered a nearly perfect environment, such a use case is not applicable for commercial use. The other long distance environments depicted a more realistic environment with objects obstructing the signals at several distances. These interferences can be seen in 5.9 as minima in the resulting polynomial regressions. A distance mapping is therefore not applicable in commercial use when using RSSI values. The CSI values in 5.11 and 5.13 showed no relevant results. The STD and CORR values ranged from 0 to 1 at all distances, as the experiment was formed at the University of Stuttgart. Human movement most likely changed the values significantly which resulted in an unusable data set of CSI values.

The results therefore show that the CSI values, which can be extracted from the ESP32-DevkitS(-R) board, are not useable for distance measurements as human and animal movements change the respective values significantly. As movements cannot explicitly be filtered out in commercial use, the given CSI values do not show promising results. RSSI can detect distances with some accuracy, depending on the the materials surrounding the given devices. Thus RSSI could be used to detect a device with a radar.

The application in 5.5 uses this exact approach. We implemented it to find a reasonable alternative for the location of devices as the results in the experiment did not precisely measure distances. The different values used in the experiment are depicted on an Android smartphone such that the user can localize a device on his own. The app has been tested on three people with 5 different test environments. The results showed that each test was successful within a reasonable time constraint.

6.1 Future Work

Further experiments can be built upon this thesis as the Radar-App in 5.14 proved to be successful in commercial use. Today's smartphones can have built-in accelometers that tracks the user's movement by measuring movements. Upon using the Radar-App, the user can track his movement while searching for the device. The outcome could result in a mapping of the building structure and the position of the devices.

A different future work could use a parameterized model, using measured values, to recognize what kind of environment is present. A test environment can be built in which new objects are put in with each measurement. The resulting CSI and RSSI values can then be used as an input for a neural network to determine the interference of specific objects in the test environment.

Notes

⁻¹https://docs.espressif.com/projects/esp-idf/en/latest/esp32/_images/esp32-devkits-v1.1-layout-front.png

⁻²https://docs.espressif.com/projects/esp-idf/en/latest/esp32/_images/esp32-devkits-v1.1-dimensions-back.png

⁻³<https://gisgeography.com/wp-content/uploads/2016/11/GPS-Trilateration-Feature.png>

Bibliography

- [BLT+06] P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, T.-C. Wang. “Semidefinite Programming Approaches for Sensor Network Localization With Noisy Distance Measurements”. In: *IEEE Transactions on Automation Science and Engineering* 3.4 (2006), pp. 360–371. doi: [10.1109/TASE.2006.877401](https://doi.org/10.1109/TASE.2006.877401) (cit. on p. 35).
- [BMPC08] K. Benkic, M. Malajner, P. Planinsic, Z. Cucej. “Using RSSI value for distance estimation in wireless sensor networks based on ZigBee”. In: *2008 15th International Conference on Systems, Signals and Image Processing*. 2008, pp. 303–306. doi: [10.1109/IWSSIP.2008.4604427](https://doi.org/10.1109/IWSSIP.2008.4604427) (cit. on pp. 18, 29).
- [Bol06] H. Bolcskei. “MIMO-OFDM wireless systems: basics, perspectives, and challenges”. In: *IEEE Wireless Communications* 13.4 (2006), pp. 31–37. doi: [10.1109/MWC.2006.1678163](https://doi.org/10.1109/MWC.2006.1678163) (cit. on p. 21).
- [DJM17] I. Dolińska, M. Jakubowski, A. Masiukiewicz. “Interference comparison in Wi-Fi 2.4 GHz and 5 GHz bands”. In: *2017 International Conference on Information and Digital Technologies (IDT)*. 2017, pp. 106–112. doi: [10.1109/DT.2017.8024280](https://doi.org/10.1109/DT.2017.8024280) (cit. on p. 42).
- [GD17] H. Ghannam, I. Darwazeh. “SEFDM over satellite systems with advanced interference cancellation”. In: *IET Communications* 12.1 (Dec. 2017), pp. 59–66. doi: [10.1049/iet-com.2017.0911](https://doi.org/10.1049/iet-com.2017.0911). URL: <https://doi.org/10.1049/iet-com.2017.0911> (cit. on p. 22).
- [GSC+18] J. González-Coma, P. Suárez-Casal, P. M. Castro, L. Castedo, M. Joham. *FDD Channel Estimation via Covariance Identification in Wideband Massive MIMO Systems*. 2018. doi: [10.48550/ARXIV.1812.00848](https://doi.org/10.48550/ARXIV.1812.00848). URL: <https://arxiv.org/abs/1812.00848> (cit. on p. 24).
- [LMSC09] K. W. K. Lui, W.-K. Ma, H. C. So, F. K. W. Chan. “Semi-Definite Programming Algorithms for Sensor Network Node Localization With Uncertainties in Anchor Positions and/or Propagation Speed”. In: *IEEE Transactions on Signal Processing* 57.2 (2009), pp. 752–763. doi: [10.1109/TSP.2008.2007916](https://doi.org/10.1109/TSP.2008.2007916) (cit. on p. 35).
- [MZW19] Y. Ma, G. Zhou, S. Wang. “WiFi Sensing with Channel State Information: A Survey”. In: *ACM Comput. Surv.* 52.3 (June 2019). ISSN: 0360-0300. doi: [10.1145/3310194](https://doi.org/10.1145/3310194). URL: <https://doi.org/10.1145/3310194> (cit. on p. 21).
- [RK21] S. P. Racharla, J. Kalaivani. “A Review on Localization Problems in Wireless Sensor Network: Algorithmic and Performance Analysis”. In: *2021 3rd International Conference on Signal Processing and Communication (ICSPC)*. 2021, pp. 31–35. doi: [10.1109/ICSPC51351.2021.9451770](https://doi.org/10.1109/ICSPC51351.2021.9451770) (cit. on p. 35).
- [SDM13] L. Schauer, F. Dorfmeister, M. Maier. “Potentials and limitations of WIFI-positioning using Time-of-Flight”. In: *International Conference on Indoor Positioning and Indoor Navigation*. 2013, pp. 1–9. doi: [10.1109/IPIN.2013.6817861](https://doi.org/10.1109/IPIN.2013.6817861) (cit. on pp. 18, 29).

- [SQA+20] D. Sánchez-Rodríguez, M. A. Quintana-Suárez, I. Alonso-González, C. Ley-Bosch, J. J. Sánchez-Medina. “Fusion of Channel State Information and Received Signal Strength for Indoor Localization Using a Single Access Point”. In: *Remote Sensing* 12.12 (2020). ISSN: 2072-4292. DOI: [10.3390/rs12121995](https://doi.org/10.3390/rs12121995). URL: <https://www.mdpi.com/2072-4292/12/12/1995> (cit. on pp. 18, 29).
- [SSL11] G.N. Shirazi, M.B. Shenouda, L. Lampe. “Second order cone programming for sensor network localization with anchor position uncertainty”. In: *2011 8th Workshop on Positioning, Navigation and Communication*. 2011, pp. 51–55. DOI: [10.1109/WPNC.2011.5961014](https://doi.org/10.1109/WPNC.2011.5961014) (cit. on p. 35).
- [STB+18] I. Strumberger, E. Tuba, N. Bacanin, M. Beko, M. Tuba. “Wireless Sensor Network Localization Problem by Hybridized Moth Search Algorithm”. In: *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*. 2018, pp. 316–321. DOI: [10.1109/IWCMC.2018.8450491](https://doi.org/10.1109/IWCMC.2018.8450491) (cit. on p. 35).
- [WSL04] Y. Wu, S. Sun, Z. Lei. “A low complexity VBLAST OFDM detection algorithm for wireless LAN systems”. In: *Communications Letters, IEEE* 8 (July 2004), pp. 374–376. DOI: [10.1109/LCOMM.2004.828187](https://doi.org/10.1109/LCOMM.2004.828187) (cit. on p. 24).

All links and footnotes were last followed on July 24th, 2022.

A Mathematical prerequisites

The algorithm focuses on the intersection of spheres, circles and point pairs. If four device positions are known, the algorithm outputs exactly one position. For three devices up to two points are returned while two devices output a circle of possible positions.

The following algorithms explain the detailed mathematical intersections necessary for the algorithm. They have explained in more detail through a froumd post of a stackexchange question¹. The link to the forum post contains the specified contents as of the 25th of July 2022. We assume in all functions that an intersection exists. In the commercial application this will always be the case, as the distances are derived through the positions of all devices. Therefore an intersection can always be computed for two devices with a given distance when the position of the devices are known. Additionally, no device positions are equal to each other, as devices cannot be located at the same three-dimensional position.

Algorithm A.1 Sphere-Sphere Intersection

```
function SPHERESPHEREINTERSECTION( $p_1, d_1, p_2, d_2$ )  
   $\vec{p} \leftarrow \vec{p}_2 - \vec{p}_1$   
   $d \leftarrow \|\vec{p}\|$   
  if  $d_1 + d_2 = d$  then  
    return  $\frac{\vec{p}}{d} * d_1$   
  end if  
   $h \leftarrow \frac{1}{2} + \frac{d_1^2 - d_2^2}{2 \cdot d^2}$   
   $\vec{c} \leftarrow \vec{p}_1 + (\vec{p}) \cdot h$   
   $r \leftarrow \sqrt{d_1^2 - h^2 \cdot d^2}$   
   $\vec{n} \leftarrow \frac{\vec{p}}{d}$   
  return circle  $\leftarrow (r, \vec{c}, \vec{n})$   
end function
```

The algorithm, displayed in A.1, returns a circle of possible positions with the input being two known position $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ and the distance of the positions to the device of which the position is unknown. The circle is returned due to it being the intersection of two spheres. In the given algorithm the circle consists of a center point c , a radius r and the normal vector of the given circle \vec{n} .

Firstly, the vector from point p_1 to p_2 is determined. The intersection is then realized by calculating the length of \vec{p} . If the calculated distance is equal to the sum of the radii d_1 and d_2 , the spheres intersect at only one point. Afterwards a variable h is declared that describes the length of a vector

¹<https://gamedev.stackexchange.com/questions/75756/sphere-sphere-intersection-and-circle-sphere-intersection>

from p_1 to p_2 , on which the center point of the circle will be located at. The resulting position determines the center of the circle which is then stored in the c variable. Afterwards the radius and the normal of the plane of the circle are calculated.

Algorithm A.2 Circle-Sphere Intersection

function CIRCLESPHEREINTERSECTION(p, d, C)

$$\vec{q} \leftarrow C.\vec{c} - \vec{p}$$

$$d_t \leftarrow C.\vec{n} \cdot \vec{q}$$

$$\vec{c}_t \leftarrow \vec{p} + C.\vec{n} \cdot d_t$$

$$\vec{s} \leftarrow \vec{c}_t - C.\vec{c}$$

if $d = C.r$ **then**

$$\quad \textbf{return } \frac{\vec{s}}{\|\vec{s}\|} \cdot d$$

end if

$$r \leftarrow \sqrt{d^2 - d_t^2}$$

$$\vec{t} \leftarrow \frac{\vec{s} \times C.\vec{n}}{\|\vec{s} \times C.\vec{n}\|}$$

$$d_p \leftarrow \|\vec{s}\|$$

$$h \leftarrow \frac{1}{2} + \frac{C.r^2 - r^2}{2 \cdot d^2}$$

$$\vec{c}_p \leftarrow C.\vec{c} + \vec{s} * h$$

$$r_p \leftarrow \sqrt{C.r^2 - h^2 \cdot d^2}$$

$$\vec{p}_1 \leftarrow \vec{c}_p + \vec{t} \cdot (-r_p)$$

$$\vec{p}_2 \leftarrow \vec{c}_p + \vec{t} \cdot r_p$$

return *pointPair* $\leftarrow (p_1, p_2)$

end function

The algorithm for the circle-sphere intersection in A.2 takes a sphere, with the center point p and radius d , and a circle C as an input. The circle consists of its center point $C.c$, radius $C.r$ and normal vector of its plane $C.\vec{n}$.

The algorithm first calculates how many units the circle cuts the sphere from the sphere's center p . As an intersection of the circle and sphere is a precondition for all intersection functions, the inequality $d_t \leq d$ will always hold.

The next step calculates the center point of a circle from the sphere, with the resulting circle of the sphere intersecting the circle C . The center point c_t of the new circle is calculated by adding $d_t \cdot C.\vec{n}$ to the sphere's center p . If the radius of the sphere is equal to the radius of the circle, the intersection returns only one intersection point. The next calculation revolves around reducing the circle-sphere intersection to a circle-circle intersection. This is done by calculating the radius of the new circle with the normal plane $C.\vec{n}$ and center point c_t . A vector \vec{t} is defined that represents the normalized tangent vector in the given plane. Afterwards, the variables h , \vec{c}_p and r_p are calculated in order to determine the two intersection points. The variables are calculated similar to the sphere-sphere intersection in A.1. The last step calculates the intersection points by using the previously temporarily defined variables and the tangent vector \vec{t} . The return value then consists of two points p_1 and p_2 .

Algorithm A.3 PointPair-Sphere Intersection

```
function POINTPAIRSPHEREINTERSECTION( $p, d, P$ )  
   $d_1 \leftarrow \|\vec{p} - P.p_1\|$   
   $d_2 \leftarrow \|\vec{p} - P.p_2\|$   
  if  $|d_1 - d| > |d_2 - d|$  then  
    return  $P.p_1$   
  end if  
  if  $|d_1 - d| < |d_2 - d|$  then  
    return  $P.p_2$   
  end if  
  return  $P$   
end function
```

The algorithm in A.3 calculates the nearest point of two points $P.p_1$ and $P.p_2$ to the edge of a sphere with center point p and radius d . The calculation is done by first calculating the distance of both points p_1 and p_2 stored in the P variable. The resulting distances are then compared by taking the absolute difference between the radius of the sphere and the calculated distances.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature