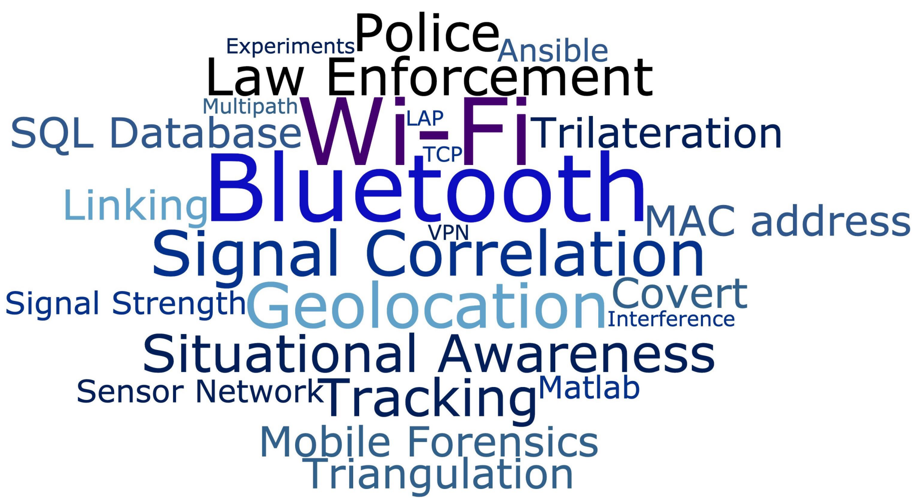
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Atle Malthe Sørenssen

# Ensuring quality of covert police work with Wi-Fi and Bluetooth technology

Master's thesis in Information Security (MISEB)
Supervisor: Professor Katrin Franke
Co-supervisor: Kyle Porter and Ivar Weider Moen
June 2021



NTNU
Kunnskap for en bedre verden

Atle Malthe Sørenssen

# Ensuring quality of covert police work with Wi-Fi and Bluetooth technology

Master's thesis in Information Security (MISEB)
Supervisor: Professor Katrin Franke
Co-supervisor: Kyle Porter and Ivar Weider Moen
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

# Abstract

The extraction and interpretation of artefacts from digital evidence is highly relevant for law enforcement. However, artefacts from wireless signals transmitted from Bluetooth and Wi-Fi devices are to a limited extent used by the police today, even though the data can contribute to investigations. In this master's thesis, we capture passive data packets from Bluetooth (Classic) and Wi-Fi and analyse the data in order to find opportunities that the police can use to increase their situational awareness in cyberspace (and real life). More specifically, we perform signal correlation that links Bluetooth and Wi-Fi MAC addresses that belongs to the same device. Combining metadata from these technologies allow us to identify the devices even if a random Wi-Fi MAC address is used. In addition, by performing geolocation, we track devices based on the received signal strength.

In the first part of this study, we designed and built a sensor network consisting of six sensors using off-the-shelf hardware and free software. The Ansible framework automated, among other things, several of the capturing processes. With the sensor network fully operational, we completed our experiments by collecting data from two groups of devices. Before starting the collection, we focused on mitigating interference and multipath propagation. Among the collected data, the most relevant data types, i.e. signal strength, MAC addresses and timestamps, were imported into a SQL database.

The primary focuses of this study have been to use the data set to link signals back to their originating device and use geolocation methods (triangulation and trilateration) to track devices. At the same time, we have explored different filtering methods to remove irrelevant data and increase accuracy.

Our results show that the most reliable signal correlation algorithm was the conversion from Bluetooth to Wi-Fi signal. This algorithm was able to link the correct pair of MAC addresses with an accuracy between 29-40 %. Among the three best signal pairs linked, it was between 43-70 % probability that the signal pair derived from the same device. Among the five best signal pairs, the accuracy increased between 57-80 %. The results from the geolocation methods showed an accuracy between 1-7 meters from the actual location and the place of estimation.

# Sammendrag

Sikring og tolkning av artefakter fra digitale beslag er svært relevant for politiet. Artefakter fra trådløse signaler som sendes fra Bluetooth og Wi-Fi enheter brukes imidlertid i begrenset grad av politiet i dag, selv om dataene kan bidra i etterforskningen. I denne masteroppgaven samler vi inn passive datapakker fra Bluetooth (Classic) og Wi-Fi, og analyserer dataene for å finne muligheter som politiet kan bruke for å øke deres situasjonsforståelse i det digitale rom (og det virkelige liv). Mer spesifikt utfører vi signalkorrelasjon for å linke Bluetooth- og Wi-Fi MAC-adresser som tilhører samme enhet. Ved å kombinere metadata fra disse tekologiene kan vi identifisere enhetene selv om en tilfeldig Wi-Fi MAC-adresse blir brukt. I tillegg sporer vi enheter basert på mottatt signalstyrke ved å bruke geolokaliseringsmetoder.

I den første delen av denne studien designet og bygget vi et sensornettverk bestående av seks sensorer ved bruk av hyllevare og gratis programvare. Blant annet benyttet vi Ansible-rammeverket for å automatisere flere av innhentingsprosessene. Med sensornettverket fullt operativt fullførte vi våre eksperimenter ved å samle inn data fra enheter som var inndelt i to grupper. Før vi startet innsamlingen fokuserte vi på å redusere interferens, samt legge til rette for at signalene skulle ha færrest mulig blokkeringer fra andre objekter i rommet. Blant den innsamlede dataen ble de mest interessante datatypene valgt og importert til en SQL-database, dvs. signalstyrke, MAC-addresser og tidsstempeler.

Det primære fokuset i denne studien har vært å bruke datasettet til å linke signaler tilbake til deres opprinnelige enhet og bruke geolokaliseringsmetoder (triangulering og trilaterasjon) for å spore enhetene. Samtidig har vi utforsket forskjellige filtreringsmetoder for å fjerne irrelevante data og følgelig øke nøyaktigheten.

Resultatene våre viser at den mest pålitelige algoritmen for signalkorrelasjon var å konvertere Bluetooth- til Wi-Fi-signal. Denne algoritmen var i stand til å linke det korrekte paret av MAC-adresser med en nøyaktighet mellom 29-40 %. Blant de tre beste signalparene som var linket var det mellom 43-70 % sannsynlighet for at signalparet kom fra samme enhet. Ved å se på de fem beste signalparene var nøyaktigheten økt til mellom 57-80 %. Resultatene fra geolokaliseringsmetodene viste en nøyaktighet mellom 1-7 meter fra den faktiske plassering til der vi estimerte.

# Acknowledgements

I would like to thank my supervisor from NTNU, Professor Katrin Franke, for her guidance and support. A special thank goes to my co-supervisors, Kyle Porter and Ivar Weider Moen which have given me advice along the way and provided me with valuable feedback.

Furthermore, I want to thank Stig Andersen and Vegard Antonsen for interesting discussions. Mathias Hansen deserves a thank you for helping me during the collection phase and Daniel Bing Andersen that contributed with his SQL skills. Last but not least, I want to thank all those who have answered my questions related to the topic, participated in my experiments and otherwise contributed to this study.

Studies take up a lot of time, and thus I would like to express my deepest gratitude to my family and friends for all your support over the past three years.

Atle Malthe Sørenssen

Stabekk, 31st May 2021

# Contents

# Figures

# Tables

# Code Listings

# Acronyms

**ACI** Adjacent-Channel Interference. 7, 35

**AFH** Adaptive Frequency Hopping. 9

**AFU** After first unlock. 79

**AP** access point. 7, 8, 36

**BFU** Before first unlock. 79

**BLE** Bluetooth Low Energy. 8, 9

**CCI** Co-Channel Interference. 7

**dBm** decibel-milliwatts. 5, 22

**E2EE** End-to-end encryption. 17

**FIFO** First-In-First-Out. 41

**FSPL** Free-Space Path Loss. 23

**GDPR** General Data Protection Regulation. 35, 43

**KDF** key derivation function. 18

**LAP** Lower Address Part. 6, 10, 31, 37, 45, 47, 83

**LOS** Line of Sight. 37

**LQ** Link Quality. 10, 12

**MAC** Media Access Control. 5, 6, 9, 10, 45

**NAP** Non-significant Address Part. 6

**NIC** Network Interface Card. 6

**OUI** Organizationally Unique Identifier. 6

**RSSI** Received Signal Strength Indicator. 5, 10, 12

**SAP** Significant Address Part. xiii, 6, 45, 46, 97

**SQL** Structured Query Language. 20, 21, 48

**SSH** Secure Shell. 17

**SSID** Service Set Identifier. 12, 36

**TCP** Transmission Control Protocol. 20, 42

**TPL** Transmit Power Level. 10, 12

**UAP** Upper Address Part. 6, 10

**VPN** Virtual Private Network. 17

# Glossary

**AFU** After First Unlock is a type of mobile forensics extraction, which is only possible to acquire if the screen lock is already entered by the user after the last reboot. The extraction could include almost 90 % of a Full File System. 46

**BFU** Before First Unlock is a type of mobile forensics extraction, which only include general information about the phone without any access to encrypted data. 46

**Elliptic-curve Diffie-Hellman (ECDH)** is a key exchange protocol that includes both private and public keys in order to send or establish a secure connection over an insecure medium. 18

**ground truth** Information that has been measured and can be linked to values obtained in the experiments. 24

**Multiple-Input Multiple-Output (MIMO)** Devices using multiple transmitters and receivers to send more data at the same time. All devices with 802.11n are MIMO compatible. 49

**triangle inequality** In a triangle, the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side. 29

**whitelisting** A filter to allow for identified devices. In this case, include only those devices belonging to the participants that had given their consent. 48

# Chapter 1

# Introduction

## 1.1 Topic covered by the project

Collecting Bluetooth and Wi-Fi data, revealing system-specific information and locating mobile phones may increase situational awareness for law enforcement. Situational awareness is defined in this master's thesis to interpret radio signals from nearby devices using Bluetooth and Wi-Fi standards to make the police more efficient and prepared before and during operational situations. Situational awareness can be, among other things, the identification of where devices are positioned, speed and direction and the tracking of devices to see movement pattern. Another feature that could help law enforcement from a technical perspective is revealing individuals behind randomised MAC addresses by combining Bluetooth and Wi-Fi metadata (through signal correlation).

This master's thesis is supervised by SDPAi (section for digital police work and innovation), which is the leading police department in Oslo within digital forensics. SDPAi has, to some extent, experience with Bluetooth and has earlier collected Wi-Fi data related to a similar project. Research from this master's thesis will hopefully help SDPAi in their process to develop new methods and techniques to comply with smart city thinking. Interpreting the data from nearby devices could potentially provide a new source of valuable intelligence to the police. Another thing the police could benefit from is the feature to link system information obtained from mobile forensics extractions to devices in a collected data set. Such information can provide filtering capabilities in a city environment that are only available to the police.

The direct focus in this master thesis will be to collect data from a network of sensor nodes and use different linking and geolocation methods to create value for the police. These methods include various tools for data collection and different techniques to filter out unwanted data. In the end, the results will be presented with graphics and statistics that show if the Bluetooth MAC address belonging to a device can be linked to the same device's Wi-Fi MAC address. Also, the geolocation results will be illustrated to show the accuracy of the tracking capabilities.

## 1.2 Keywords

Covert police work, Situational awareness, Bluetooth sniffing, Wi-Fi scanning, Identifying devices, Mobile forensics, Signal correlation, Geolocation tracking

## 1.3 Problem description

In order to have a safe society, we need the police to maintain public order, protect citizens, prevent crimes and investigate them. Based on these social tasks, there is a common understanding that the police must adapt to new technology and utilise new methods in the cyber domain. While this is the case, some laws and regulations restrict covert methods when it affects people's privacy and the use of these methods if nothing criminal is suspected. This study will look into possibilities and not the limitations regulated by law.

In general, police work is often linked to situations where there is a lack of a good overview and situations that require good planning in advance of operations. Another situation the police is facing is to protect important persons and buildings. Common to all these situations is that best practices do not include the collection and interpretation of information from mobile devices that use Bluetooth and Wi-Fi. With respect to covert methodologies, these new methods may be used to conduct a greater overview of the situation with tactical intelligence on which decision-makers can act. Imagine each police unit as a mobile sensor that feeds the operation centre with live intelligence from nearby devices. Another suggestion is mobile equipment placed in backpacks to collect Bluetooth and Wi-Fi data in close vicinity of the target. Such methods would be beneficial for covert operations to locate nearby devices as a supplement to already existing techniques. After all, different scenarios are dependent on the environment and thus the need to use different technologies or enrich each other. In addition, investigators could also benefit from this information to build better timelines that can confirm or disprove if a specific device was in proximity of a criminal act. If these data are linked to mobile forensics extractions, the police would also have the ability to identify the mobile phone owner. On this basis, the relationship between signals and devices may be essential to look into.

## 1.4 Research questions

To solve the problem statement described in 1.3, the main research question is formulated as such:

- **How can the police use Bluetooth and Wi-Fi data for tracking and identification in covert operations?**

In order to answer the question mentioned above, the following sub-questions must be further analysed:

1. What useful information can be collected passively from Bluetooth and Wi-Fi data?
2. Given that both Bluetooth and Wi-Fi are enabled, how can one find that these signals originate from the same device?
3. Which algorithms can be considered best to link Bluetooth and Wi-Fi signals originating from the same device?
4. Which geolocation algorithms can be considered best to track devices using signal strength from Bluetooth and Wi-Fi data?
5. Can signal interference be a problem while collecting data?
6. What technical challenges may arise when the police collect data from Bluetooth and Wi-Fi?
7. How should irrelevant and misleading data be filtered out?

## 1.5 Justification, motivation and benefits

Being a special investigator working with digital forensics, we constantly search for new methods and capabilities to interpret artefacts and exploit vulnerabilities in order to reveal the true story. We are always focusing on data integrity and the correct use of the chain of custody.

The police are dependent on adapting to new technology to investigate criminal cases in the best way possible. Therefore, the overall desire is to make the police more efficient by using more artefacts from a forensics perspective. The underlying motivation for this project is to contribute to this process to make sure that the police is aware of the valuable information that can be extracted from Bluetooth and Wi-Fi data. Hopefully, this project will enlighten these possibilities and perhaps help covert police work become more efficient with a greater level of situational awareness in cyberspace (and real life). To emphasise the results, we suggest concrete use cases below the discussion part in the last chapter.

## 1.6 Planned contributions

In contrast to studies performed by Kolberg [1], Groba [2] and Chilipirea et al. [3], which interpreted Wi-Fi packets (probe requests) that were rarely transmitted, this master's thesis looks at active Wi-Fi data streams (TCP packets) that are more often transmitted from devices. Also, Bluetooth packets will be analysed.

The goal of this master's thesis is to give the police a new source of intelligence by increasing situational awareness with Bluetooth and Wi-Fi technology. This intelligence will be obtained by interpreting metadata from data packets to geolocate devices and performing signal correlation of Bluetooth and Wi-Fi signals. Data packets from Bluetooth and Wi-Fi often contain metadata such as MAC address, device name, vendor and signal strength. Such information combined will hopefully give the police a greater level of situational awareness out in the field.

Overall, this master's thesis will present an overview of the architecture used, how data were collected, and the methods used to filter the data. Also, the algorithms used to determine if a Bluetooth and Wi-Fi signal originates from the same device and the methods used for geolocation will be presented. The main scripts developed in Matlab will be attached in Appendix D, while relevant use cases will be suggested in the discussion part below Section 7.1.4.

## 1.7 Thesis structure

In the following chapters, the master's thesis is structured as follows:

**Introduction:** In the first chapter, the reader is given context to the topic. The introduction is also where concrete tasks performed in the thesis are described.

**Background:** This chapter provides the theoretical basis of the thesis, such as terms related to Bluetooth, Wi-Fi, different methods and algorithms. Related work is also included in this chapter.

**Methodology:** The third chapter presents the methods that were used to collect data, the filtering options chosen and those geolocation methods and signal correlation algorithms tested. The chapter also gives an overview of how the sensor network was built, which tools were implemented and the equipment used.

**Experiment setup:** Contains in-depth descriptions of the different scenarios and how the experiments were performed. This chapter also focuses on what information each scenario and experiment would provide.

**Pre-processing and data analysis:** Takes the reader thoroughly through the processing and analysis part, focusing on valuable data from the data set. Also, this chapter dives into some methods that need further explanations.

**Discussion and conclusion:** The last chapter contains a discussion part, the strengths and limitations of the study, possible use cases and findings. Further, the research questions are accounted for and concluded. Finally, there are some suggestions to further work to may be carried out.

**Appendix:** Additional information and attachments such as the complete equipment lists, whole system architecture and Matlab code are found in the appendices.

# Chapter 2

# Background

This chapter includes a technical background and brief descriptions of studies related to the research questions mentioned in Section 1.4. Over the years, several studies have examined how Bluetooth and Wi-Fi data can be collected and analysed with the overall purpose of tracking. These studies are also deemed relevant for this master thesis by being a starting point for further assessment and analysis.

- Technical background
- Related work

## 2.1 Technical background

### 2.1.1 Signal strength

One of the most important parameters acquired in the data collection is the received signal strength measured in decibel-milliwatts (dBm). This is an absolute value representing the received power in mW (milliwatts) on a logarithmic scale (1 mW = 0 dBm)[4]. Unlike the Received Signal Strength Indicator (RSSI), which measures the received signal on a relative scale, and which varies greatly between different manufacturers [4], signals measured in dBm relates to the same scale. Normally, Wi-Fi signals vary between -70 and 0 dBm, while Bluetooth signals vary between -80 and 0 dBm [5][4]. These variations, of course, depend on how much power the signal is transmitted with. Values closer to zero are stronger than lower values. Research performed by Longo [6] shows that signals measured in dBm correlate better with distance than using RSSI. In this master's thesis, the software capturing Wi-Fi data (TCPdump) and Bluetooth data (Ubertooth) included the absolute value of the received signal measured in dBm.

### 2.1.2 MAC address

Another important parameter is the Media Access Control (MAC) address, which is a unique identifier assigned to its hardware for correct addressing in communication. The MAC address, also referred to as a physical address, is primarily generated by the manufacturer and assigned to all Network Interface Card (NIC) such as the NIC for Bluetooth and Wi-Fi. People who are interesting in operational security may want to keep their public MAC address concealed. The safety focus is because the MAC address can be linked to the owner's identity and tracked based on the device presence [8]. Figure 2.1 illustrates how a 48 bits Bluetooth MAC address is structured. The Non-significant Address Part (NAP) and Upper Address Part (UAP), together forming the Organizationally Unique Identifier (OUI), is vendor-specific, while the Lower Address Part (LAP) is device-specific. The latter can be used to identify devices, even though OUI is not discovered. The combination of UAP and LAP is called the Significant Address Part (SAP). Even though the MAC address in Figure 2.1 relates to a Bluetooth device, the structure of a Wi-Fi MAC address is similar, but only divided into OUI (first 24 bits) and a unique NIC identifier (last 24 bits).

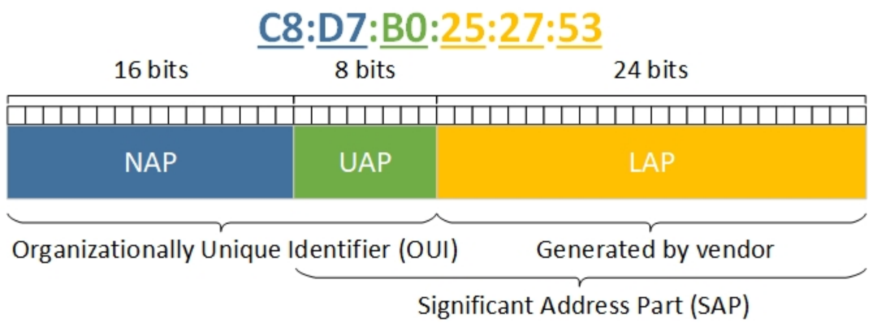

C8:D7:B0:25:27:53
16 bits
8 bits
24 bits
NAP
UAP
LAP
Organizationally Unique Identifier (OUI)
Generated by vendor
Significant Address Part (SAP)

**Figure 2.1:** Public Bluetooth MAC address (BD_ADDR) from a Samsung device

From a security and privacy perspective, the MAC address for Wi-Fi and Bluetooth (only Low Energy) are commonly randomised in order to hide the address and to prevent tracking of the device [7][8]. In this context, a distinction is made between public and random MAC addresses. The public address is the original MAC address given by the manufacturer, while the random (also called private) MAC address is randomised and used openly for communication [9].

When it comes to mobile forensics, the public MAC address related to Bluetooth and Wi-Fi can be presented and extracted in forensic tools even though the device is locked. This accessible information could give law enforcement an advantage in a smart city setting, where the MAC addresses from mobile extractions could be correlated with data from out in the field (e.g. covert police work and demonstrations). If the data coincide, it is a high probability that this device belonging to that person was present in, e.g. the demonstration.

### 2.1.3 Wi-Fi and capturing data frames

*Wi-Fi* is a wireless communication technology that consists of several protocols based on the IEEE 802.11 standard [10]. Wi-Fi enables wireless connectivity that often allows devices to communicate with each other or with the internet through an access point (AP) or router. The radio waves are transmitted and received in the Gigahertz range, generally in 2.4 GHz and 5-6 GHz, depending on which Wi-Fi protocol is used. An overview of different Wi-Fi protocols that are released since 1999 are listed in Table 2.1. The AP in this thesis was configured to use the Wi-Fi 4 protocol with the frequency option set to 2.4 GHz.

| Wi-Fi generation | IEEE standard | Released | Max data rate | Frequency |
|---|---|---|---|---|
| Wi-Fi 1 | 802.11a | 1999 | 54 Mbps | 2.4 GHz |
| Wi-Fi 2 | 802.11b | 1999 | 11 Mbps | 5 GHz |
| Wi-Fi 3 | 802.11g | 2003 | 54 Mbps | 2.4 GHz |
| Wi-Fi 4 | 802.11n | 2009 | 600 Mbps | 2.4/5 GHz |
| Wi-Fi 5 | 802.11ac | 2014 | 1.3 Gbps | 5 GHz |
| Wi-Fi 6 | 802.11ax | 2019 | 10-12 Gbps | 2.4/5 GHz |

**Table 2.1:** Different Wi-Fi protocols with generation names [11]

Out of 14 different Wi-Fi channels, there are only 13 that are available in Europe. Three of those are so-called non-overlapping channels with 5 MHz in between to mitigate for Adjacent-Channel Interference (ACI), where devices transmit with overlapping frequencies and thus "talk over each other". Concerning the possibility for interference with Bluetooth devices, the frequency was in this thesis set to channel 11 in the AP configuration. This choice could increase the Co-Channel Interference (CCI) between devices using the same frequency in the same area, such as Wi-Fi devices connected to the same access point or between Wi-Fi and Bluetooth devices [12]. However, it is best practice to accept some CCI in order to avoid ACI [12]. The frequency associated with channel 11 is shown in Figure 2.2.

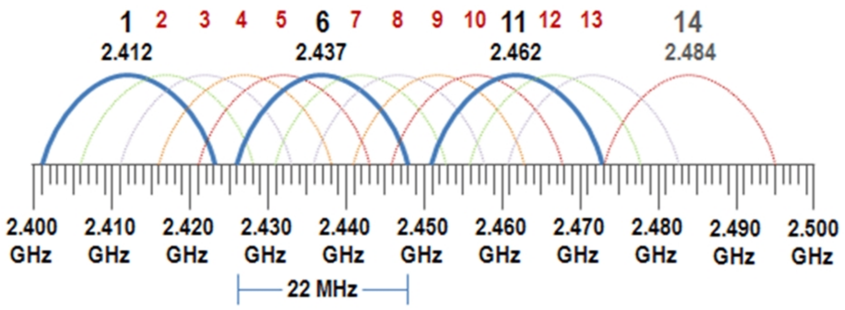


**Figure 2.2:** Wi-Fi frequencies and channels [12]

### Capturing Wi-Fi data frames

Wireless network traffic can be passively captured with the use of third party application such as TCPdump. One of the methods is to let TCPdump set the WLAN interface in promiscuous mode in order to capture all network traffic from devices that are associated with the same AP [13]. This configuration means that network traffic that is supposed to arrive at a specific device also is captured by the manipulated WLAN interface. For each data frame (802.11) that are captured with TCPdump, additional information called radiotap header is encapsulated in the original frame [14]. In this case, the radiotap header includes supplementary information, e.g. the received signal.

### 2.1.4 Bluetooth, discovering and capturing data frames

*Bluetooth* is a wireless communication technology standard intended for short-range distances. Bluetooth consists of two non-compatible standards, and both included in newer smartphones. Bluetooth Classic (BR/EDR) is the oldest standard, which is currently used for streaming music, while Bluetooth Low Energy (BLE) is designed with the intention of increased security and lower power consumption. The Classic standard has improved less over the years in comparison with the newer standard, which lately has been greatly updated in Bluetooth 5. Although the newest Bluetooth 5.2 standard, introduced at Consumer Electronics Show (CES) in 2020, includes a game-changing standard for transferring audio data (LE Audio) with a new audio codec called LC3 (Low Complexity Communication Codec), most devices supporting this standard will not hit the marked before later in 2021 [15]. Because devices are lacking support, Bluetooth Classic will still be used for audio transfer for several years. The rows highlighted in green in Table 2.2 show the variety of Bluetooth version's used among the devices in this master's thesis.

| Versions | Released | Important Bluetooth features |
|---|---|---|
| 1.0, 1.0B | 1999 | First edition |
| 1.1 | 2002 | IEEE Standard 802.15.1, non-encrypted channels |
| 1.2 | 2005 | Improved speed, faster connection and discovery, Adaptive Frequency Hopping (ADH), backward compatible |
| 2.0 + EDR | 2004 | General improvements, enhanced data rate (EDR) |
| 3.0 + HS | 2009 | Higher speed, enhanced power control |
| 4.0 | 2010 | Bluetooth Low Energy (LE) introduced, dual-mode, support for generic attribute profile (GATT), increased security for LE, range and connectivity improvements |
| 4.1 | 2013 | Indirect IoT device connection, interference improvement |
| 4.2 | 2014 | Security improvement to Bluetooth LE |
| 5 | 2016 | Range and speed improvement |
| 5.1 | 2019 | Range and speed improvement |
| 5.2 | 2020 | LE audio with a new standard Bluetooth audio codec called L3C, broadcast audio feature |

**Table 2.2:** How the Bluetooth standards have evolved since 1999 [16]

To limit how much power a Bluetooth antenna transmit, both standards have regulated the power consumption into three main (one extra for BLE) classes [17]. Smartphones that use Bluetooth for general communication are operating within the power range of class two and can transmit with a maximum of 4 dBm. Figure 2.3 shows an overview of these power classes.

| **Class** | **Output power** | **Range range** | **Sample devices** |
|---|---|---|---|
| Class 1 | 100 mW (+20 dBm) | 100 m | IoT, industrial |
| Class 1.5 (BLE) | 10 mW (+10 dBm) | 30 m | Beacons, wearable |
| Class 2 | 2.5 mW (+4 dBm) | 10 m | Mobile devices, smart card |
| Class 3 | 1 mW (0 dBm) | 1 m | Bluetooth adapters |

**Table 2.3:** Power classes of Bluetooth devices [17]

Bluetooth is coexisting with Wi-Fi in the 2.4 GHz unlicensed industrial, scientific and medical (ISM) frequency band. The Classic standard leverages an Adaptive Frequency Hopping (AFH) approach, meaning that Bluetooth devices are trying to mitigate interference by using spread spectrum techniques [18]. In a point-to-point connection, two Bluetooth devices are using the AFH technique to rapidly change frequencies among 79 channels based on a secret pattern they have agreed upon [19]. Figure 2.3 show all the channels associated with Bluetooth, each spaced 1 MHz apart. The three non-overlapping Wi-Fi channels are also illustrated in the same frequency range.

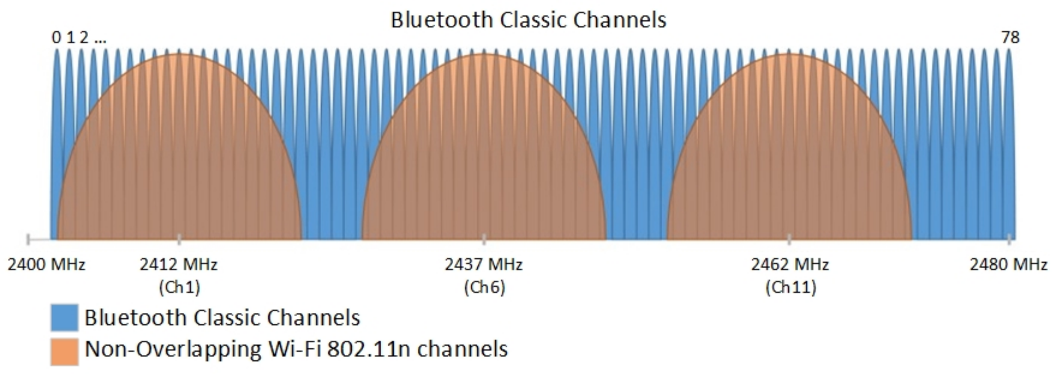


**Figure 2.3:** Bluetooth coexisting with Wi-Fi in the 2.4 GHz ISM band [19]

### Bluetooth Classic discovering and capturing

One of the Bluetooth Classic shortcomings is the lack of MAC address randomisation, which BLE have implemented. Instead, Bluetooth Classic use a defence mechanism to remain non-visible, even though Bluetooth is enabled [20]. The security feature means devices can avoid being listed when someone is performing an inquiry scan to search for Bluetooth devices. However, this feature is exploited with equipment and software provided by Ubertooth. Ubertooth is able to passively capture and demodulate signals in the 2.4 GHz ISM band. Each of these Bluetooth packets (pseudo-header for BR/EDR) contain the unique LAP which could be used to identify a device. Other parameters included in the packets are the received signal strength in dBm and the timestamp of when the signal was received. Another feature in Ubertooth is the survey mode, which can determine the UAP by capturing packets from the same device over time. The latter is more time consuming, only revealing one UAP at a time.

BlueZ is another Linux-based software that officially supports the core Bluetooth layers and protocols. The primary tool within Bluez is the host controller interface tool (HCI tool) which can send commands to Bluetooth devices in order to fetch information such as RSSI, Link Quality (LQ) and Transmit Power Level (TPL) [6]. However, fetching this information is only possible during an active connection and when the full Bluetooth MAC address is known. To overcome this, it is possible to constantly send L2ping requests that the device will need to answer. These requests will keep the connection active while the HCI tool is fetching, e.g. RSSI. To acquire the MAC address of nearby devices, a discovery service called Blue Hydra in combination with the Ubertooth antenna would automatically reveal public MAC addresses over time [21]. Our tests (performed in April 2020) with Blue Hydra shows that a brand new Samsung with Android 10 over time would reveal its MAC address while streaming music to a Bluetooth device. However, studies show that Blue Hydra fails to collect RSSI values [22]. On this basis, Ubertooth is in this master's thesis chosen to collect Bluetooth Classic data frames.

## 2.2 Related work

This section will contain research that may be relevant to the main research question: *How can the police use Bluetooth and Wi-Fi data for tracking and identification in covert operations?*. Although some of the research is limited to either Wi-Fi or Bluetooth, the principles may still be relevant for this master's thesis. The various research papers chosen relate to these topics:

- Tracking devices in public with Bluetooth Classic and Wi-Fi
- Individual tracking with Wi-Fi
- Bluetooth Low Energy
- Technical challenges
- Signal correlation
- Geolocation methods
- Filtering methods

#### Tracking devices in public with Bluetooth Classic and Wi-Fi

Research performed by Bai et al. [23] focused on how devices can be tracked in public. When performing their research, they created a sensing system to determine how many passengers were using public transport. The experiment lasted over five days and consisted of series of bus and tram journeys. To later verify the passenger numbers, they also manually counted the people. By counting mobile devices using Bluetooth and Wi-Fi, the sensing system correlated passenger loads relative to ground truth information. Various filters were used to remove noise, such as devices held by people waiting at bus stops or devices passing by. By using the filters, the results become more accurate and could be correlated to the actual number of passengers [23].

In another study performed by Schauer et al. [24], Bluetooth and Wi-Fi data were used to identify crowd densities and pedestrian flows at an airport. By comparing the collected data with the actual number of people checking in through the airport security, it was possible to check the accuracy of the filters used. Furthermore, the study shows that the amount of trackable Bluetooth devices is less in comparison to the number of boarding pass scans. However, the Wi-Fi density estimations are more accurate. Schauer et al. concluded that both Bluetooth and Wi-Fi would be useful to approximate crowd densities in airports [24].

### Individual tracking with Wi-Fi

In 2019, Tsai et al. [25] performed a study in which location data from Wi-Fi were used to reveal the direction of a suspect. The direction was predicted by way of using six ordinary sensors in addition to multiple auxiliary observation nodes connected to each sensor. If three auxiliary nodes detected the same device, the information about the location, signal strength and MAC address was uploaded to a server. Regardless of the exact location of the suspect, the system would manage to calculate the suspect's direction [25].

Another study related to indoor tracking has been performed by Kolberg [1]. In order to track devices, Kolberg tested different geolocation methods on a Wi-Fi data set consisting of probe requests. Both triangulation and trilateration are covered in her study, which seeks to give law enforcement an increased situational awareness by tracking individual devices and interpret crowd densities. Her results show that it is possible to locate and pinpoint individuals within approximately five meters. In addition, the study includes several aspects of interpreting Wi-Fi data that are relevant in this thesis.

### An interesting tool using Bluetooth Low Energy

Turning the focus over to the Bluetooth Low Energy (BLE) protocol, a study by Hexway [26] reveals that Apple's mobile devices leak status messages by passively sniffing BLE traffic. Information such as Wi-Fi status (on/off), screen status (lock screen, home screen, off, calling), and iOS version can all be detected using a tool called Apple Bleee. The tool would also detect password requests that the phone transmits in some situations, which can be used to guess the original phone number [26]. The research mentioned above may be helpful for the police to quickly get an overview of Apple products in an area or obtain vital information regarding screen status that is very useful from a mobile forensics perspective.

**Technical challenges**

There is a lot of previous research within the field of Bluetooth and Wi-Fi data. However, the studies have mainly focused on using anonymised data and not specifically on identifying mobile devices. While this anonymised data can be helpful in some scenarios, the police will often need to identify devices uniquely. At this point, technical challenges arise. Due to new privacy legislation (e.g. General Data Protection Regulation – GDPR), technology companies have implemented security features in their operating systems to comply with the legislation.

Ansley [7] and Ryan et al. [27] have studied these security features related to Wi-Fi on Android and iOS. Their research shows that mobile devices are set to default using MAC address randomisation when they are not associated with a network. When associating these devices to a known network, their public MAC addresses will be revealed in the advertising probe requests [7]. According to our tests (April 2020), this is currently true for iOS (13.3.1) and Android 9. However, devices with Android 10 (and higher) and iOS 14 (and higher) are using MAC address randomisation even though the devices are associated with a network [28][29]. In addition, own tests performed in October 2020 shows that as long as the SSID name remains the same, devices using Android or iOS will keep their random Wi-Fi MAC addresses over time. The random MAC address is found in the Wi-Fi menu related to each SSID. This finding is a bit contrary to what apple says about Wi-Fi MAC randomisation - "it can't be used to persistently track a device by passive observers of Wi-Fi traffic" [30]. Despite this, Ryan et al. [27] mention the ability to track devices by creating fake access points with which the devices are familiar. This method could lure the target devices to transmit probe requests containing the true Wi-Fi MAC addresses.

With regards to Bluetooth security, almost the same challenges will apply. A study performed by Becker et al. [31] shows that even if the security features in Bluetooth are optional, the manufacturers often implement randomisation of MAC addresses. Davies et al. [20] have performed research on how to monitor non-discoverable Bluetooth Classic devices. Results show that off-the-shelf hardware such as Ubertooth One can discover information from devices regardless of whether the discoverable mode is turned on or off. The study shows that there are approximately 4.7 times as many Bluetooth devices in non-discoverable mode as in discoverable mode [20].

Similar research performed by Longo [6], shows that Bluetooth parameters such as RSSI, LQ and TPL can be obtained by establishing an active connection with L2ping and by sending commands to the devices based on their MAC addresses. Even though Longo [6] used old devices without MAC address randomisation in his research, this could still be possible by using Blue Hydra to reveal the true MAC address. However, our tests show that it is challenging to hold several active L2ping connections to multiple devices while requesting RSSI values. Because of this, this method was discarded midway through the master's thesis. In this specific field, the literature appears to be weak. Nevertheless, the methods and hardware above-mentioned are essential in this thesis to encounter security challenges to identify devices uniquely.

### Signal correlation

One of the research questions in this project is related to the linking of Wi-Fi and Bluetooth signals transmitted from the same device. As random Wi-Fi MAC address is more standard these days, linking these signals would increase the possibility of uniquely identify devices. Research done by Longo [6] is in this case relevant, as he managed with high probability, through the use of RSSI values, to pair Wi-Fi and Bluetooth signals coming from the same device. The experiment was performed indoors with six sensors. Several algorithms were tested in order to separate the signals and pair those originating from the same device. However, this research did not encounter the same security challenges which this project must overcome. However, signal correlation of Bluetooth and Wi-Fi signals, as Longo performed, in combination with geolocation tracking methods mentioned by Groba and Chilipirea et al., would potentially give law enforcement a greater situational awareness.

In order to link Bluetooth and Wi-Fi data, the parameters inside the data packets need to be analysed. Longo assumed that the signals coming from the same device would look different but should correlate due to the same representation of the distance between device and sensor [6]. Of five algorithms, one, in particular, stood out. This algorithm was the conversion from RSSI to distance. The study shows that the RSSI was highly dependent on the transmitting chipset [6]. To overcome this challenge, Longo created a logarithmic regression line of each device. On this basis, the RSSI values were converted more correctly into the distance. Further on, he used Euclidean distance to compare and link the signals back to the same device. Moreover, by increasing the number of sensors from four to six, the results became more accurate. Of the five closest values (top-k approach), the signals could be linked with 100 % accuracy [6]. Longo's preliminary experiments would also be relevant in this thesis to see if various chipsets are transmitting with different output power. Concerning covert operations, this would be not easy to perform in a real life scenario. Pursuant to Longo [6], it is possible to use on-the-fly devices hidden at known locations to create a trustworthy relation between the distance and the received signal strength.

### Signal interference

Another research question in this master's thesis is how the signals will affect the accuracy of signal strength during the collection phase. Both Wi-Fi and Bluetooth use the same 2.4 GHz frequency band (ISM), which could lead to interference. Research from Pei et al. [32] shows that Bluetooth devices do have an influence on Wi-Fi positioning when the technologies coexist in the same environment. However, bringing a Bluetooth device in a connected state will enable a mechanism called adaptive frequency hopping (AFH). According to Pei et al., the AFH mechanism will reduce the interference between Bluetooth and Wi-Fi. On this basis, the Bluetooth devices would be distributed nicely to different channels if they, e.g. stream music to a connected Bluetooth headset.

### Geolocation methods

There are three main geolocation methods considered in this master's thesis. One of these is called Time of Arrival (TOA). TOA is calculating the distance based on the time difference from when a signal is sent until it is received and the speed of light (constant in vacuum) [33]. Due to the speed the signal uses to propagate, a deviation of one nanosecond in the TOA measurement will affect the distance by 0.3 meters [33]. Since the experiment is to be conducted indoors, the signals will likely be reflected by the environment, which affects the propagation time from sender to receiver. Based on this information from research performed by Schauer et al., the ToA method will not provide the accuracy needed in this project. The inaccuracy also applies to the method called Time Difference of Arrival (TDOA), which calculates the distance between one device and two reference points based on the time difference received at the reference points [34]. As Schauer et al. points out, TDOA requires high-end equipment and very accurate time synchronisation to obtain accurate results [33].

Another method that can measure the distance is Angle of Arrival (AOA). This method takes advantage of calculating the angle from the received signal based on the time difference between multiple antenna elements [35]. Unfortunately, as Sarshar describes in his research, these multi-array antennas are costly and suffer from complex indoor environments with land-of-sight as a requirement [35]. Additionally, this method requires a high degree of time accuracy (nanoseconds). Features introduced in Bluetooth 5.1 (BLE) show that AOA could be used to locate devices more accurately in the future [36].

The third geolocation method calculates the distance between the sender and the receiver using received signal strength. This method is a proven method performed in several studies already mentioned, such as Kolberg, Longo, Schauer et al., and Bai et al. Distance estimation using signal strength will be further explained in Section 3.6.

### Filtering methods

Collecting Wi-Fi and Bluetooth signals will require methods to filter out unimportant and misleading data. Reducing this data will contribute to smaller and more manageable log files and make the information more reliable and actionable for law enforcement. Relevant research by Kurkcu and Ozbay [37], Longo [6], Groba [2] and Chilipirea et al. [3] mention several filtering methods to apply for this goal. One of these methods is time-based filtering used by Groba to exclude devices based on probe requests that appear more than once at each sensor. A similar method performed by Kurkcu and Ozbay [37] were to calculate the wait time of each device. With this information, they could remove devices discovered for longer than some minutes but also shorter than some seconds. Another filtering method mentioned by Groba [2] and Chilipirea et al. [3] is called distance-based filtering. The goal of this method is to filter out all irrelevant data and signals of low quality. This method is similar to what Longo mentions in his research, where all corrupted probe requests were removed. Longo also mentions a method for filtering out devices that were passing by. His method ignored those MAC addresses related to less than ten probe requests [6].

While most of the research papers above-mentioned have focused on collecting data from a public tracking perspective, this master's thesis seeks to fill in the gaps to identify devices and gain individual tracking capabilities. The overall goal is to increase situational awareness for law enforcement with Bluetooth and Wi-Fi data. To achieve this, signal correlation based on Wi-Fi and Bluetooth combined with different tracking methods will be tested and compared. It will be essential to collect enough data (Wi-Fi and Bluetooth) that contains the same fundamental parameters. In addition, it will be necessary to utilise several filtering techniques, as previous research mention, to obtain more reliable and actionable information. The research carried out by Kolberg [1], and Longo [6] have been two of the most inspiring related work.

# Chapter 3

# Methodology

The quantitative approach in this master's thesis consisted of collecting data through Bluetooth and Wi-Fi in several experiments and use it to geolocate mobile devices and perform signal correlation. Such methods could increase situational awareness for law enforcement to become an analysis tool supplement for decision-makers. In order to describe the methods used, this chapter is divided into these main sections:

- System architecture
- Data filtering and cleaning methods
- Managing the collected data
- Geolocation methods
- Trilateration and triangulation
- Linking algorithms

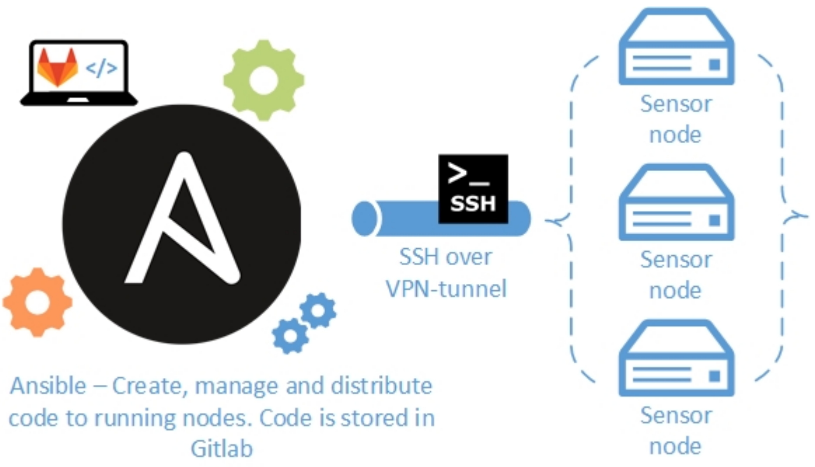
## 3.1 System architecture

Before collecting data, the system architecture was planned, configured and tested. An overview of the final architecture is shown in Figure 3.1. In this phase, important functionalities and parameters needed more attention, such as received signal strength, timestamps and MAC addresses. In addition, correct time synchronisation on the equipment collecting the data was needed to secure data integrity.

GitLab was used as the main platform for source code and collaboration. In good cooperation with our project partner, the tools listed in Section 3.2 were implemented and written in the Ansible scripting language. Ansible made it possible to administrate and distribute tasks such as updating all sensor nodes with new functionality from GitLab and perform time synchronisation. Also, Ansible made it easy to start and stop the data collection from all six sensors simultaneously during the data collection. To ensure a secure connection, Secure Shell (SSH) over Virtual Private Network (VPN) was established on all nodes. This gave End-to-end encryption (E2EE). A software called Wireguard was selected as the VPN-tunnel because it was simple, free and uses different encryption functionality such as

(i) symmetric key encryption with Chacha20[1], (ii) Elliptic-curve Diffie-Hellman (ECDH)[1] for key exchange, (iii) encryption hash functions including Blake2 and SipHash24 and (iv) a key derivation function (KDF) called HKDF [38].

Due to a large amount of data, the way of saving log files was changed during the data collection. First, the data were sent back to the controlling node but later saved locally on each sensor node. Completing the system architecture took longer than expected but was, in return, a system that ensured easy and efficient data collection. Good help from the project partner was appreciated.



</>
SSH
SSH over VPN-tunnel
Sensor node
Sensor node
Sensor node
Ansible – Create, manage and distribute code to running nodes. Code is stored in Gitlab

**Figure 3.1:** Architecture of the sensor network

## 3.2 Capabilities - Tools

To be able to capture data packets, each sensor node was configured equally and included tools listed in Table 3.1. These tools were carefully tested before they were implemented as services and could be started and stopped individually with ansible commands.

| **Tools** | **Technology** | **Area of use** |
|---|---|---|
| Ubertooth | Bluetooth Classic | Sniffing data packets |
| TCPdump | Wi-Fi | Capturing data packets |
| Blue Hydra | Bluetooth | Unveil public MAC addresses |
| HCI tool | Bluetooth | Retrieve Bluetooth information from devices |
| L2Ping | Bluetooth | Ping devices |
| Apple Bleee | Bluetooth Low Energy | Retrieve screen status from Apple devices |

**Table 3.1:** Different tools installed on each sensor node

[1]Explained in the list of glossary

## 3.3 Equipment

The equipment used was hosted and funded by the project partner. New equipment that was needed had a cost of approximately NOK 20,000. Among the equipment, it was simple computers (Raspberry Pi's), Wi-Fi antennas, Bluetooth dongles and Ubertooth devices. These antennas were not calibrated equally and thus held a consumer-grade with the disadvantages it entails. During the experiments, additional equipment was used, such as a GoPro camera for recording movements, a Wi-Fi router to enable internet and a laptop node for administrating the sensors. Another laptop was used as a time source for comparing timestamps between data packets and the recordings. This time source needed to be visible in the recordings during the experiments. The smartphones subjected to additional testing over time, grouped as "known devices", were collected from friends, family and work. There was a total of 10 known devices in addition to ten Bluetooth audio devices. For the complete list of equipment, see appendix A.

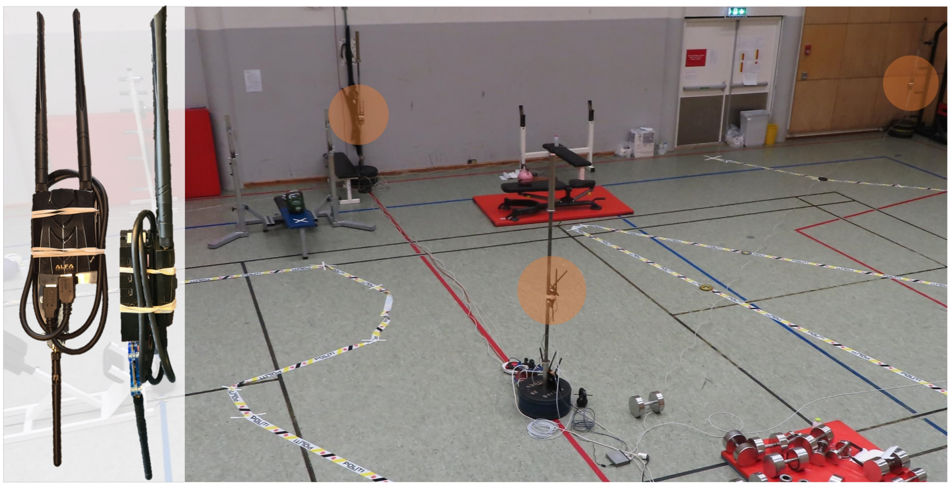


**Figure 3.2:** Sensor nodes (S4, S5 and S7) in the controlled environment

## 3.4 Data cleaning and filtering methods

One of the research questions stated in Section 1.4 was to analyse the collected data to find filtering options and apply them to those methods that could give the most accurate results. In this section, all the methods used will be covered, while Section 5.3 in the pre-processing and data analysis chapter explains in detail how some of these were utilised.

#### Unwanted observations

In order to prevent alteration of the relevant data, just necessary cleaning was performed before importing the data to SQL. After all, further filtering and cleaning methods could be performed in Matlab. The first method involved filtering out irrelevant data. This filter implies reducing the raw data logs only to include data packets from relevant devices (devices included in the project experiments). The same approach for eliminating irrelevant devices was performed by Kurkcu and Ozbay in their initial filtering process [37]. By applying this method to the log files related to TCPdump, the file size was significantly reduced, which resulted in faster filtering in the database.

#### Structural error

When the collected data were imported into the SQL database, the information was closely examined to see if something was missing. Some information related to one experiment was quickly identified as missing (not imported correctly) and was immediately corrected in an improved import script during this examination. This typo could have had a major impact on the result if it had not been discovered. Such errors can often appear in data transferring or during data acquisition. [39].

#### Time-based filtering

Another method is to filter data based on time. In combination with recordings and a time source, this method was used in this master's thesis to filter data from a specific experiment or filter a specific time slot when a participant moved from one location to another. All data outside the chosen time frames would then be filtered out. Other research, such as Groba [2], utilised time-based filtering in his demonstration scenario to exclude devices that appeared more than once at each sensor. This approach would not have worked in this thesis, as Groba's sensors were aligned in a straight line with longer distances between each sensor to find the direction of devices.

#### Time compression

In contrast to research performed by Groba [2], who concludes that probe requests are only showing a fraction of the actual attendance, this thesis has collected active TCP packets. There are advantages and disadvantages to the different approaches, but in this master's thesis, the amount of data has not been an issue. In fact, there has been so much Wi-Fi data down to milliseconds that one method used was time compressing. Time compression means that all data within the exact second is compressed into an average time. When we applied this method to the Wi-Fi data, it improved the speed when querying data from the SQL database to Matlab because it was compressed into fewer rows. In addition, it became easier to handle the data in Matlab based on seconds rather than milliseconds.

### Distance-based filtering

One obvious method used in this master's thesis is a combination of distance-based filtering and unwanted out-liners. This filter means reducing the amount of data by filtering out received low-quality signal values, which consist of too weak, too strong or values that research shows are irrelevant. Kurkcu and Ozbay [37], Groba [2] and Chilipirea et al. [3] do mention this method in their research to filter out non-relevant signal values. Our distance-based filtering method implemented technology-specific thresholds to limit the received signal strength. These thresholds prevented the methods from calculating positions outside the room or calculating longer or shorter distances than realistic. As Chilipirea says, it is vital to set a threshold because "there is no one-size-fits-all" solution due to different equipment and sensor structures [3]. Further details of how this method was utilised are given in Section 5.3.

### Missing data

One mechanism that needed to be implemented in the Matlab script was in regards to missing data. Several if-statements were able to discover null values or values outside the policy during the execution of the script. These statements were necessary in order to unveil missing data and for the script to continue without interruptions. Two ways of dealing with missing data are by imputing data based on other similar observations or by dropping observations with missing or wrong values [39]. In this thesis, only a few missing data have been identified. In these cases, the device or values associated with that device have been ignored. Section 5.3 will give further information about missing data.

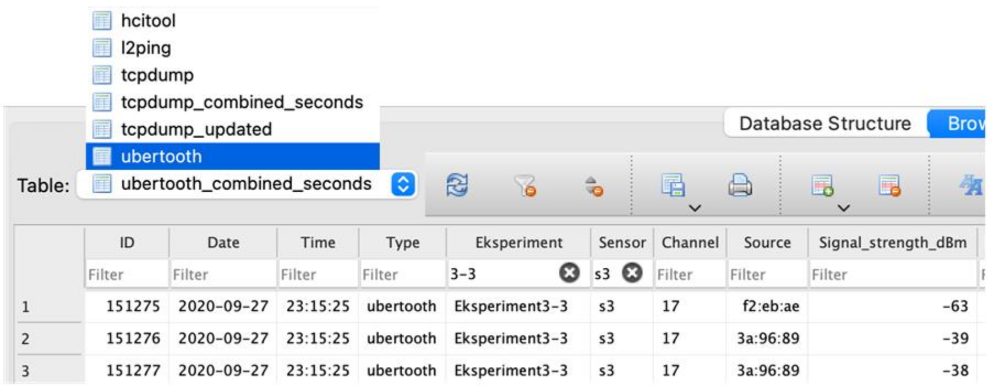
## 3.5 Managing the collected data

### SQL database

Following the filtering and cleaning methods mentioned in Section 3.4, the data was imported into a database supporting Structured Query Language (SQL). This activity was performed by the project partner based on an import plan with an overview of data types and column names. Each tool was given its table. Having all the data available in different tables inside a database made it possible to find data from different experiments more quickly. It also provided a good overview for following up the methods performed in Section 3.4.

### Matlab code

Due to several mathematical calculations, Matlab was chosen as the scripting language to implement methods for geolocation and signal correlation. The Matlab code, which is presented in Appendix D, consists of two scripts. The first script connects to the SQL database and filters the desired data from different tables (shown



| | ID | Date | Time | Type | Eksperiment | Sensor | Channel | Source | Signal_strength_dBm |
|---|---|---|---|---|---|---|---|---|---|
| | Filter | Filter | Filter | Filter | 3-3 | s3 | Filter | Filter | Filter |
| 1 | 151275 | 2020-09-27 | 23:15:25 | ubertooth | Eksperiment3-3 | s3 | 17 | f2:eb:ae | -63 |
| 2 | 151276 | 2020-09-27 | 23:15:25 | ubertooth | Eksperiment3-3 | s3 | 17 | 3a:96:89 | -39 |
| 3 | 151277 | 2020-09-27 | 23:15:25 | ubertooth | Eksperiment3-3 | s3 | 17 | 3a:96:89 | -38 |

**Figure 3.3:** Imported data in the SQL database with different tables

in Figure 3.3) based on SQL queries. Further on, this information is stored in new tables created in Matlab. In script two, these tables are used in order to calculate the chosen geolocation or linking methods. Code Listing 3.1 and 3.2 show the connection between Matlab and SQL and show how Matlab is querying data.

**Code listing 3.1:** Connection between SQL and Matlab

```
dbfile = ('Atle_Loggfiler.db'); % Current database file
conn = sqlite(dbfile); % Creates a database connection to "dbfile"
```

**Code listing 3.2:** Example of SQL query from Matlab

```
% SQL query with options like experiment number, signal strength, etc
pre_sqlquery = ['SELECT',time_format,',Eksperiment,Source,
round(',signal_strength_type,',4),Sensor,Filepath,Combined,
Sum_signalstrength FROM ',current_type,' WHERE "Eksperiment" LIKE
',current_experiment,' '];
        pre_data = fetch(conn,pre_sqlquery); % Saves data from query to pre_data
```

## 3.6 Geolocation methods

This section describes the location methods which have been used. The focus will lay on finding algorithms that estimate the distance between a device and the sensor to locate devices and understand the propagation path loss.

### Distance estimation using signal strength

To estimate where a signal is coming from, the distance between the transmitting device and the receiving sensor is essential. This distance can be expressed with a common value included in Wi-Fi and Bluetooth data frames, which is the signal strength measured in decibel-milliwatts (dBm). The signal strength is the basis for all algorithms within this master's thesis because it can be used together with other information to estimate the distance between a transmitting device and the receiving one [40].

The first step towards estimating the distance is to know how much a signal is reduced when propagating through air. By using a power link budget, the received signal ($P_R$) at the sensor can be subtracted by the transmitted signal ($P_T$) from the device. The result of the subtraction, called Free-Space Path Loss (FSPL), is the signal reduction that has been influenced by distance or external factors. The measured signal reduction can be calculated as follows:

$$FSPL_M = P_T - P_R \tag{3.1}$$

Another way of expressing the reduction in signal strength over a distance was presented by Friis [40]. His transmission equation was based on the ratio between the received power ($P_r$) and the transmitted power ($P_t$). This expression was further based on the characteristics (Aperture) of the transmitting antenna ($A_T$), the receiving antenna ($A_R$), the antenna-separation distance (d) and the wavelength ($\lambda$) shown in 3.2.

$$\frac{P_r}{P_t} = \frac{A_r A_t}{d^2 \lambda^2} \tag{3.2}$$

A more convenient way of expressing the free-space path loss formula is in decibels (dB). Assuming the antennas are isotropic (no fixed directivity which disregards the aperture), it is possible to derive equation 3.2 into this equation (in terms of frequency) [41]:

$$FSPL(dB) = log_{10}\left(\left(\frac{4\pi df}{c}\right)^2\right) \tag{3.3}$$

where:

$log_{10}$: Logarithm with base 10
$d$: Distance between transmitter and receiver in meter
$f$: Transmitted frequency in MHz
$c$: Speed of light in vacuum (approximately $300{\cdot}10^6$ m/s)

At this point, we can input $FSPL_M$ from equation 3.1 to the left side of equation 3.3. Since we already know the frequency, it is possible to solve the equation based on the distance (d). The equation changes into:

$$d = 10^{\left(\frac{FSPL_M - 20log(f) - 20log\left(\frac{4\pi}{c}\right)}{20}\right)} \tag{3.4}$$

Example: If a phone plays music and transmits the audio to a Bluetooth device with $P_T$ = 4 dBm and the received signal strength at the sensor node is measured to be $P_R$ = -55 dBm, then the power link budget according to equation 3.1 will show a signal reduction equal to $FSPL_M$ = 4-(-55) = 59 dBm. The distance is further calculated in equation 3.5. Due to frequency hopping in the Bluetooth standard, the frequency factor is solved by summarising the upper and lower Bluetooth frequency channel and divide it by two, which equals 2441 MHz (mean value).

$$d = 10^{\left(\frac{59-20log(2441)-20log\left(\frac{4\pi}{300}\right)}{20}\right)} = 8.7m \tag{3.5}$$

Equation 3.4 will in this master's thesis be used in both the geolocation methods and every linking algorithm (except normalisation). It is important to note that accuracy can be affected by interference, different apertures and the angle at which the devices are held [40]. However, an estimation of the distance between the sender and a sensor node may be good enough for law enforcement.

## 3.7 Trilateration or triangulation using signal strength

### 3.7.1 Trilateration

One method to locate devices is by way of using trilateration. The basis of this method is knowing the distance between at least three sensors and the device to determine the relative position of the device [42]. An illustration of this method is shown in Figure 3.4. Only four out of six sensors located in each corner were selected (S1, S2, S5 and S6) to reduce the complexity of this method. Still, with three sensors calculating the location of the device, there are four different compositions of the sensors (equation 3.8 - 3.11). These compositions could be compared in order to find which is the most accurate in device tracking.

While ground truth[2] is obtained during the experiments, the challenge arises in a real-life scenario where the actual position of a device will need to be calculated without knowing the distance. A potential solution would be to estimate the distances with a surveillance camera. Another approach, which may be sufficient for law enforcement, is to convert the received signal strength into the distance with the Friis equation (3.4) and at the same time be aware of deviations. This approach will be the basis for this method.

To proceed with calculations of trilateration, some algebra knowledge is needed. The standard formula for a circle is:

$$r_i{}^2 = (x - x_i)^2 + (y - y_i)^2 \tag{3.6}$$

where:

$r_i$: Distance from centre of sensor $i$ to mobile device
$x$ & $y$: The coordinates to where the mobile device could be located
$x_i$ & $x_i$: x and y coordinates where sensor $i$ is positioned

[2]Explained in the list of glossary

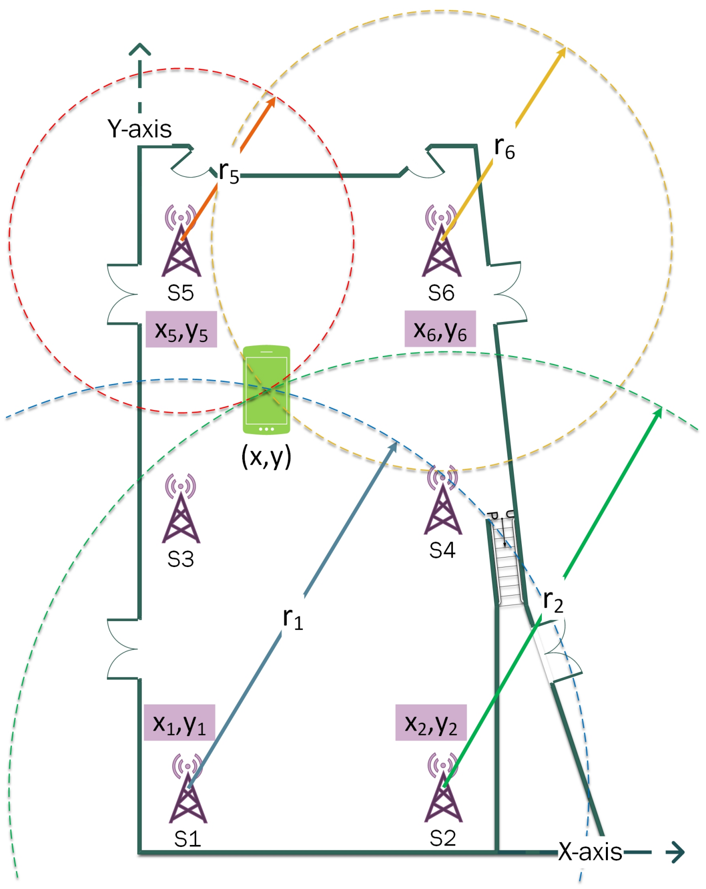


**Figure 3.4:** Illustrates the concept of trilateration using algebra

**Step one** uses formula 3.6 to create four equations, one for each sensor. In a best-case scenario, each of these equations intersects in the same coordinates where a device is located. Thus, the values of x and y are ideally. The goal of this calculation is to find a set of the equation for x and y.

$$r_1{}^2 = (x - x_1)^2 + (y - y_1)^2 \qquad \text{(Equation 1 - Sensor 1)}$$

$$r_2{}^2 = (x - x_2)^2 + (y - y_2)^2 \qquad \text{(Equation 2 - Sensor 2)}$$

$$r_5{}^2 = (x - x_5)^2 + (y - y_5)^2 \qquad \text{(Equation 3 - Sensor 5)}$$

$$r_6{}^2 = (x - x_6)^2 + (y - y_6)^2 \qquad \text{(Equation 4 - Sensor 6)}$$

**Step two** expands the squares into a generic equation (description is given in equation 3.6:

$$r_i{}^2 = x^2 - 2x(x_i) + x_i^2 + y^2 - 2y(y_i) + y_i^2 \tag{3.7}$$

**Step three** is about getting linear expressions by subtracting the second equation from the first, the fifth equation from the second, the sixth from the fifth and the first from the sixth. These subtractions will eliminate the squared unknowns ($x^2$ and $y^2$), making the expressions easier to deal with and allow using linear algebra. The equations are further rewritten where x and y are moved to the left side. After moving x and y, all the symbols on the right side of the equation are known. Also, the equations are simplified by substituting the coefficients with letters.

$$r_1^2 - r_2^2 - x_1^2 + y_1^2 - x_2^2 + y_2^2 = 2x(-x_1 + x_2) + 2y(-y_1 + y_2) \qquad \text{(Equation 1)}$$

$$\Rightarrow 2x(-x_1 + x_2) + 2y(-y_1 + y_2) = r_1^2 - r_2^2 - x_1^2 + y_1^2 - x_2^2 + y_2^2$$

$$\Rightarrow Ax + By = C \qquad \text{(Equation 1 substituted)}$$

where:

$A$: $2(-x_1 + x_2)$
$B$: $2(-y_1 + y_2)$
$C$: $r_1^2 - r_2^2 - x_1^2 + y_1^2 - x_2^2 + y_2^2)$

$$r_2^2 - r_5^2 - x_2^2 + y_2^2 - x_5^2 + y_5^2 = 2x(-x_2 + x_5) + 2y(-y_2 + y_5) \qquad \text{(Equation 2)}$$

$$\Rightarrow 2x(-x_2 + x_5) + 2y(-y_2 + y_5) = r_2^2 - r_5^2 - x_2^2 + y_2^2 - x_5^2 + y_5^2$$

$$\Rightarrow Dx + Ey = F \qquad \text{(Equation 2 substituted)}$$

where:

$D$: $2(-x_2 + x_5)$
$E$: $2(-y_2 + y_5)$
$F$: $r_2^2 - r_5^2 - x_2^2 + y_2^2 - x_5^2 + y_5^2$

$$r_5^2 - r_6^2 - x_5^2 + y_5^2 - x_6^2 + y_6^2 = 2x(-x_5 + x_6) + 2y(-y_5 + y_6) \qquad \text{(Equation 3)}$$

$$\Rightarrow 2x(-x_5 + x_6) + 2y(-y_5 + y_6) = r_5^2 - r_6^2 - x_5^2 + y_5^2 - x_6^2 + y_6^2$$

$$\Rightarrow Gx + Hy = I \qquad \text{(Equation 3 substituted)}$$

where:

$G$: $2(-x_5+x_6)$
$H$: $2(-y_5+y_6)$
$I$: $r_5^2-r_6^2-x_5^2+y_5^2-x_6^2+y_6^2$

$$r_6^2-r_1^2-x_6^2+y_6^2-x_1^2+y_1^2=2x(-x_6+x_1)+2y(-y_6+y_1) \qquad \text{(Equation 4)}$$

$$\Rightarrow 2x(-x_6+x_1)+2y(-y_6+y_1)=r_6^2-r_1^2-x_6^2+y_6^2-x_1^2+y_1^2$$

$$\Rightarrow Jx+Ky=L \qquad \text{(Equation 4 substituted)}$$

where:

$J$: $2(-x_6+x_1)$
$K$: $2(-y_6+y_1)$
$L$: $r_6^2-r_1^2-x_6^2+y_6^2-x_1^2+y_1^2$

**The final step** is about solving a system of equations which contains information from three sensors (trilateration). This means that two of the equations below must be solved with respect to x and y in order to find the coordinates.

- Equation 1: $Ax+By=C$
- Equation 2: $Dx+Ey=F$
- Equation 3: $Gx+Hy=I$
- Equation 4: $Jx+Ky=L$

Solving the system with linear algebra based on equation one and two:

$$Sensor(1,2,5): x=\frac{CE-BF}{AE-BD}, y=\frac{CD-AF}{BD-AE} \tag{3.8}$$

Solving the system with linear algebra based on equation two and three:

$$Sensor(2,5,6): x=\frac{FH-EI}{DH-EG}, y=\frac{FG-DI}{EG-DH} \tag{3.9}$$

Solving the system with linear algebra based on equation three and four:

$$Sensor(5,6,1): x=\frac{IK-HL}{GK-HJ}, y=\frac{IJ-GL}{HJ-GK} \tag{3.10}$$

Solving the system with linear algebra based on equation four and one:

$$Sensor(6,1,2): x=\frac{LA-KC}{JB-KA}, y=\frac{LA-JC}{KA-JB} \tag{3.11}$$

The equations in 3.8-3.11 are further implemented into the Matlab script where every letter is replaced with their respective expressions. An ideal scenario is already shown in Figure 3.4, where all the sensors convert the measured signal strength into the distance and use one of the systems set above to calculate the exact location of the device. In real life, the sensors will receive different signal strength due to external factors and interference. Therefore, it is interesting to see which of the equations above (3.8-3.11) gives the most accurate results when the devices are both at rest, but also when they are moving away from sensor 1-2 towards the sensor 5-6. This is explained more in detail in the chapter about experiment setup below scenario 1 (4.3.2) and scenario 2 (4.3.5).

### 3.7.2 Triangulation

Principles related to trigonometry will apply in this thesis when mobile devices are to be located through triangulation. In fact, when a phone is transmitting a signal, two receiving sensors can form a triangle in which the adjacent and opposite side can be calculated. This process assumes that the received signal is converted to distance (free-space path loss formula 3.4) to further calculate the angles in the triangle. Figure 3.5 illustrates how triangulation is performed.

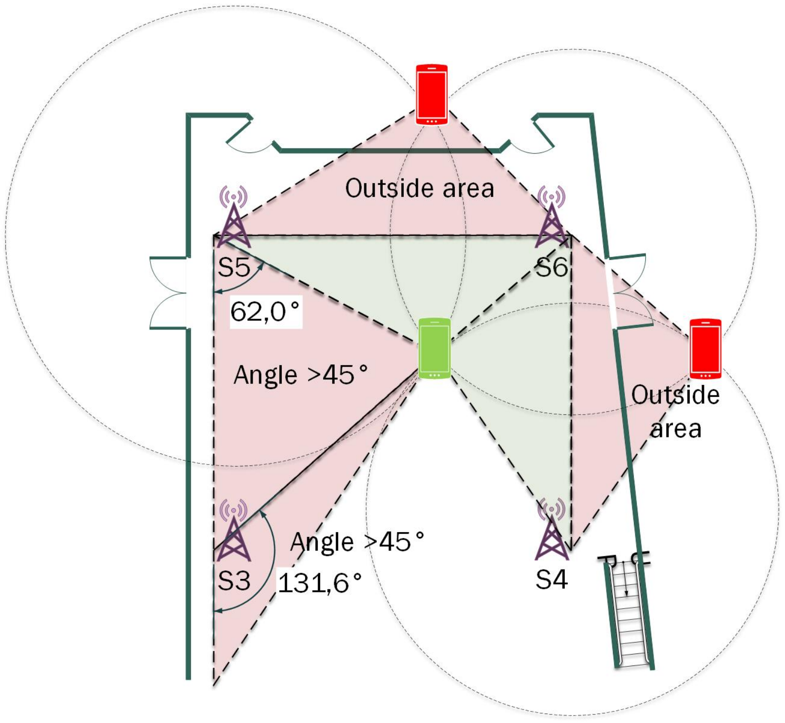


**Figure 3.5:** Illustrates the concept of triangulation using trigonometry

To exclude locations outside the environment, restrictions have been implemented in the Matlab script, which, for example, only care about angles between 0 and 45 degrees. In this case, the red devices and the red areas in Figure 3.5 will be ignored.

First, the six sensor nodes formed four medium and large triangles as seen in

Figure 3.6. Along the way, the number of triangles was increased with additional four small triangles, with a total of eight triangles. These extra small triangles occur along the long wall, where also the big triangles occur. The medium triangles occur on the short wall. Furthermore, a distinction is made between whether the angle of the triangle is calculated from the left sensor (A perspective) or the right sensor (B perspective). All these triangles give a total of 16 possibilities for calculating the device location. The possibilities are, of course, limited by the restricted angles that have been set in each triangle. Figure 3.6 shows the overlapping sectors with some of the restrictions given.

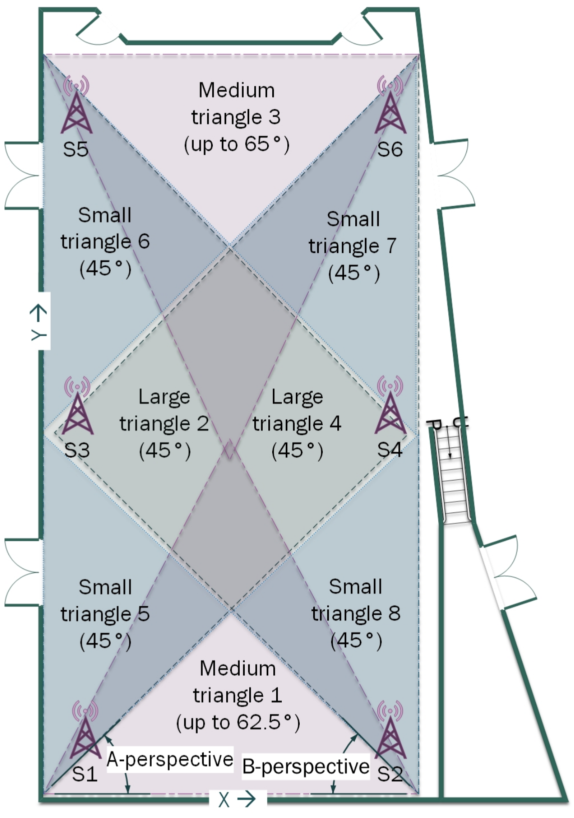


**Figure 3.6:** Illustrates all triangles, their overlapping sectors and restricted angles

To prevent the calculated distances from falling outside the area and to prevent triangle inequality [3], several if-statements have been implemented. Various values related to the if-statements are tested with the data sets to obtain as accurate locations as possible. These checks related to distance are stated below:

- Small triangle: $10.40 \leq d_{tot} \leq 15.7$ & $d < 11.195$
- Medium triangle: $11.195 \leq d_{tot} \leq 30.7$ & $d < 15.35$
- Large triangle: $20.80 \leq d_{tot} \leq 30.7$ & $d < 15.35$

[3]Explained in the list of glossary

where:

$d_{tot}$: Both distances (in meter) from sensors to device added
$d$: Distance (in meter) between sensor and device

To calculate the adjacent and opposite side in a triangle (A-perspective), such as in Figure 3.7, the law of cosines and the Pythagorean theorem need to be used.

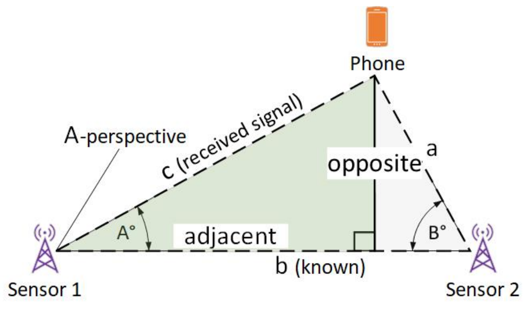


**Figure 3.7:** Adjacent and opposite length need to be calculated

**Step one** is to find the angle of A with the law of cosines. The received signal values of a and c are converted into the distance with the free-space path loss formula 3.4:

$$a^2 = b^2 + c^2 - 2bc \cdot \cos(A) \Rightarrow \cos(A) = \frac{b^2 + c^2 - a^2}{2bc} \tag{3.12}$$

where:

$a$ & $c$: Received signal from phone converted into distance
$b$: Known distance between Sensor 1 and Sensor 2
$A$: The angle from Sensor 1 (A-perspective) to the phone
$B$: The angle from Sensor 2 (B-perspective) to the phone

**Step two** uses the previous angle and distance c to find the length of the adjacent side of the right triangle:

$$cos(A) = \frac{b^2 + c^2 - a^2}{2bc} = \frac{adjacent}{hypotenuse}, (hypotenuse = c) \tag{3.13}$$

$$adjacent = c \cdot \cos(A) \tag{3.14}$$

**The last step** uses the Pythagorean theorem to find the length of the opposite side of the right triangle:

$$opposite^2 = c^2 - adjacent^2 \tag{3.15}$$

$$opposite = \sqrt{c^2 - adjacent^2} \tag{3.16}$$

Finally, these values can be plotted in a Cartesian coordinate system as x and y values: (adjacent, opposite). Even though some triangles in Figure 3.6 are rotated, the adjacent and opposite side is converted into the same coordinate system.

## 3.8 Linking algorithms using signal strength

This section is about linking Bluetooth and Wi-Fi signals that are originating from the same device. The goal is to associate the identifiable Lower Address Part (LAP) captured in Bluetooth signals with the Wi-Fi MAC address that could be randomised. In order to do so, there are several signal correlation methods or linking algorithms to choose between based on the signal strength. The algorithms that are applied in this master's thesis are:

1. Normalisation
2. Signal strength to distance conversion
3. Bluetooth signal to Wi-Fi signal conversion
4. Triangulation
5. Trilateration

### 3.8.1 Euclidean distance

The algorithms mentioned above are the first step in finding the most similar Bluetooth-Wi-Fi signal pair to highlight which phone these signals originate from. The second step, which is used at the end of all the algorithms, is about finding the most similar vectors. This calculation is achieved with a distance metric called Euclidean distance. Euclidean distance will calculate the straight line between two (or more) points, that is, between a sensor and a mobile device [43, p. 145]. The devices are likely to be in a straight line without too many obstacles that may reflect the signal. Hopefully, the correct pair of signals will combine important system information to identify a specific device, which had been difficult with just metadata from one of these signals. To later verify the correctness of the algorithms, system information related to Bluetooth, Wi-Fi and random MAC address was collected from each device involved in the experiments.

With the different composition of sensors (either 4 or 6 sensors), the Euclidean distance for points given by Cartesian coordinates in higher dimensions is calculated as follow:

$$d(w,b) = \sqrt{(w_1 - b_1)^2 + (w_2 - b_2)^2 + .. + (w_i - b_i)^2 + .. + (w_n - b_n)^2} \tag{3.17}$$

where:

$w_i$ and $b_i$: are the Wi-Fi and Bluetooth values represented by the signal strength at sensor $i^{th}$. The composition of $i$=1,2,..,n, where n = 4 or 6 tells

the structure of sensor nodes. With i = 1,2,5,6, four of the sensor nodes are chosen.
$d(w, b)$: will be close to zero if the two vectors are more similar and will be greater than zero if the vectors are different.
$w$ and $b$ : two points representing the Cartesian coordinates of a device.

### 3.8.2 Signal to distance

Converting signal strength to distance is one of the primary algorithms used in this master's thesis. In this context, each data set consisting of Wi-Fi and Bluetooth signals were converted into distances with the use of the free-space path loss formula 3.4. Further, Euclidean distance (3.17) calculated the most similar vectors. These vectors represent the line between sensor $i$ and where the device should be located based on signal strength.

### 3.8.3 Normalisation

Perhaps the simplest algorithm used is the process of normalising the data. This algorithm does not require any pre-calculations or analysis of the devices in advance but may, in turn, be less accurate. The process of normalisation involves the adjustment of measured signal strength on different scales to restructure on a common scale [44]. By normalising both the Bluetooth and Wi-Fi data to a scale between 0 and 1, it would be possible to find similar values due to the same representation of distance [6]. The normalisation of each data set is performed with this equation:

$$N_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{3.18}$$

where:

$N_i$: is the $i^{th}$ normalised data related to device $i, ..., 10$
$x_i$: signal strength value received by one of the six sensors ($x_i$, .., $x_6$)
$x_{Min}$: Lowest signal strength measured among the six sensors
$x_{Max}$: Highest signal strength measured among the six sensors

The normalised data is then compared with Euclidean distance trying to link the MAC addresses together.

### 3.8.4 Bluetooth signal to Wi-Fi signal conversion

Another algorithm is the Bluetooth signal conversion to Wi-Fi signal. Because the relation between the technologies are linear, as Longo [6] noticed, the values from Bluetooth is possible to be converted to Wi-Fi and vice versa. Functions from the regression lines obtained during the preliminary experiment (described in chapter 4) are used to convert the signals. In Figure 3.8, one of these lines is represented by the mean value of all ten known devices. The line is plotted in Microsoft Excel, while the equation is further used in Matlab for the conversion. All ten known devices do have their regression line, making it more accurate, while the unknown devices will use the mean regression line as the generic conversion.

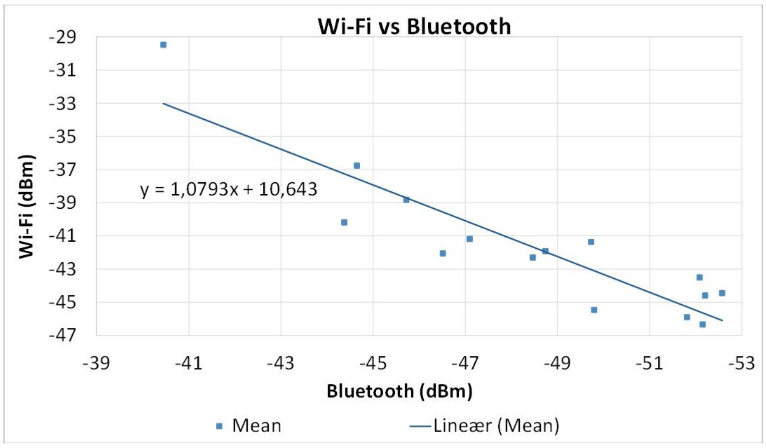


**Figure 3.8:** Plot of the mean linear regression line from all ten known devices

Having the Bluetooth signals converted into Wi-Fi signals, these two data sets were compared with the Euclidean distance to find the most similar pair.

## 3.8.5 Trilateration

In Section 3.7.1, trilateration was described. This method is also tested as a candidate for linking the pair of Bluetooth and Wi-Fi signals back to the originating device. Because of four different compositions of sensors that give slightly different coordinates associated with the same device, the mean value of these coordinates was used. To find similar signal pairs in this method, we used the Euclidean distance. This equation is based on a two-dimension formula with the vectors representing the distance of the received Bluetooth and Wi-Fi signals:

$$d(w,b) = \sqrt{(w_x - b_x)^2 + (w_y - b_y)^2} \tag{3.19}$$

where:

$w_x$ and $w_y$: mean $x$ and $y$ Wi-Fi signal strength represented by the distance.
$b_x$ and $b_y$: mean $x$ and $y$ Bluetooth signal strength represented by the distance.
$d(w,b)$: will be close to zero if the two vectors are more similar and will be greater than zero if the vectors are different.
$w$ and $b$: two points representing the Cartesian coordinates of a device.

## 3.8.6 Triangulation

Lastly, triangulation is tested as a linking algorithm. The algorithm will use the same approach as the location method in Section 3.7.2. However, since different triangles give slightly different coordinates associated with the same device, the mean value of these coordinates was used. The Euclidean distance in Equation 3.19 was used to compare signals and link those that are most similar.

## 3.9 Methodology flowchart

To summarise the methodology chapter, different phases of the master's thesis is shown in Figure 3.9 as a flowchart. First, the system architecture was developed, including an encrypted network, various nodes, and software to acquire Bluetooth and Wi-Fi data packets. Further on, the necessary equipment was ordered and assembled. The next phase consisted of network configuration and software testing to ensure that the data was correctly acquired, stored and easily retrievable. When the system was completed, different scenarios and experiments were prepared in a collection plan. It was this plan that was closely followed during the data collection. After the data was acquired, the raw data logs needed to be verified before the cleaning and filtering methods were performed. At this stage, the Wi-Fi logs were heavily reduced in size before importing the data to SQL. This reduction made the SQL queries against the database significantly faster. The next phase consisted of data analysis before scripting in Matlab began. The Matlab phase included the implementation of filtering options, linking methods and geolocation methods. After the scripting phase was completed, the scripts were used to perform linking and geolocation calculations. Finally, the results were analysed before a conclusion was given.

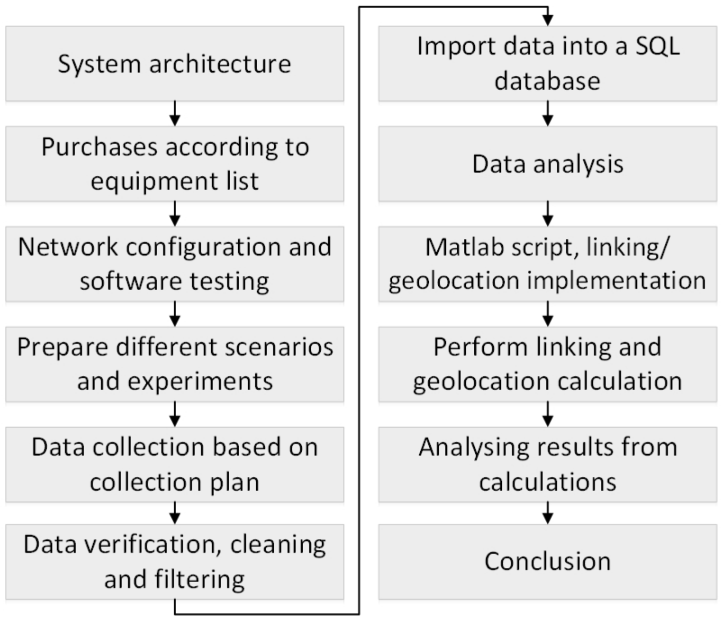

System architecture
Purchases according to equipment list
Network configuration and software testing
Prepare different scenarios and experiments
Data collection based on collection plan
Data verification, cleaning and filtering
Import data into a SQL database
Data analysis
Matlab script, linking/ geolocation implementation
Perform linking and geolocation calculation
Analysing results from calculations
Conclusion

**Figure 3.9:** Different phases in the master's thesis illustrated as a flowchart

# Chapter 4

# Experiment setup

This chapter describes the collection phase of our experiments in which we collected Bluetooth and Wi-Fi data from different phones in different scenarios, modelling different contexts. In order to collect these data packets, a collection plan was created. This plan described in detail when, where, what, why, who and how the data should be collected. Three days were used to prepare and complete in total twelve experiments. The experiments included preliminary work on the group of known devices to understand signal strength at different distances. These phones are described in experiment one through five. Ten colleagues were invited to carry out the same experiments to make them more realistic. These devices are the group of unknown devices described in experiment 6.1, 6.2, 6.3, 7.1 and 7.2.

In general, the devices would connect to a Wi-Fi network and stream a specific YouTube video while listening on a Bluetooth audio device. During this session, a controlling node gave commands to start/stop acquiring data. The Wi-Fi network was configured to only allow for 2.4 GHz with the fixed frequency channel 11. Usually, the frequency channel is set to auto, but using a fixed frequency variable in the Friis equation would make the calculated distance more precise. Also, using a non-overlapping Wi-Fi channel, as described in Section 2.1.3, the potential risk of Adjacent-Channel Interference (ACI) would decrease. However, this can be seen as a limitation of our work.

## 4.1 Participants

To maintain the legal aspect of collecting personal information such as phone MAC address, the participants in the group of unknown devices were requested to provide their consent. The consent form contains information about the participation, the purpose of the experiments and the person responsible for the data collection. It also provided information about laws related to the General Data Protection Regulation (GDPR). Because of GDPR, information requested in the consent form and the data collection were kept on a specific laptop with additional backup to an external hard drive. Both were encrypted with the state of the art encryption. Also, in accordance with the GDPR, neither personal information nor specified device information is possible to link to the thesis. The information was only used to compare data to ground truth. In addition to the information requested, the participants gave their phones' random Wi-Fi MAC address associated with the SSID on the access point (AP) and the public MAC address on their Bluetooth device. This was not specified in the consent form but was necessary for a general overview and since one scenario included the use of a random MAC address. The participants were also given a fixed number that described which location they should be in during the static experiments and another number that described in which order they should begin to start walking during the dynamic experiments. This information could later be linked together with time and video recordings showing the actual location of the devices. Both the project partner and a police lawyer approved the consent form before the invitation was sent.

## 4.2 Environment

The environment used for the experiments was a gym hosted by the project partner. The sensor nodes were located in the corner of the room (S1-S6), while one sensor node (S7) and the access point (AP) were located in the middle of the room. All of the sensors had the same hardware (see equipment list in Appendix A) and were placed at a similar height from the floor. The orientation of the antennas was either pointing down or up, making it almost vertical to the floor plane.

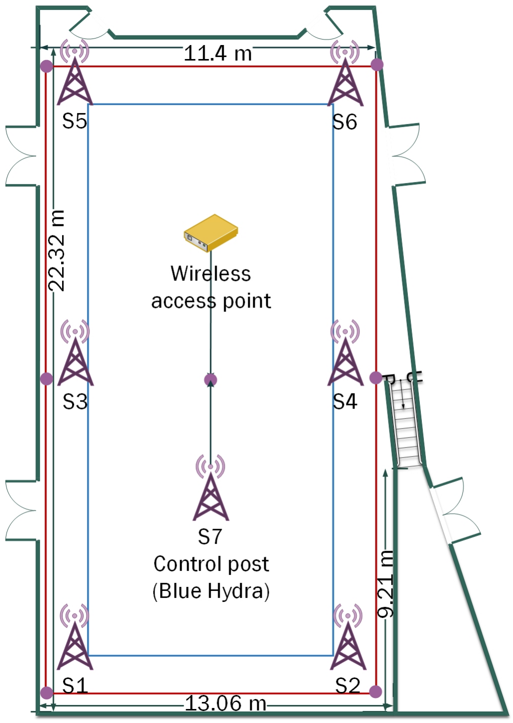


**Figure 4.1:** Illustration of the environment including sensor nodes and AP

In Figure 4.1, the purple circles illustrate where the sensor nodes were located, while the red rectangle illustrates a pre-defined rectangle that corresponds with the location of the sensors. Most of the experiment's activity were performed within the blue rectangle, also marked on the floor. Exercise equipment was already set out in the room. This equipment was not taken away during the collection, making it more realistic than if the room had been empty. In addition, the presence of ten participants during experiment six through seven could also increase the possibility of signal absorption, making the results more inaccurate [45].

## 4.3 Scenarios and Experiments

Six different scenarios were created in order to collect the desired amount of data. These scenarios consisted of preliminary work and experiments that the group of unknown and known devices would perform. The overall goals for scenario 1-4 were to link Bluetooth data with Wi-Fi data and estimate distances to geolocate devices in the room by the received signal strength. Initially, system information related to the participants' devices were unknown. So, when the participants arrived, the software called Blue Hydra (introduced in Section 2.1.4) began to scan for nearby devices in order to reveal the public Bluetooth MAC address or the Lower Address Part (LAP). The result of this scan was later used to relate with the LAP in the Ubertooth logs.

The collection plan was followed throughout the experiments, but minor changes were made in some scenarios for practical reasons. These changes are described in the scenarios below. During experiment two through five, an associate helped with setup and preparation.

### 4.3.1 Preliminary scenario

The preliminary scenario, which is illustrated in Figure 4.2 consisted of two experiments. The first experiment was intended to collect basic data from 10 smartphones to plot signal strength as regression lines at different distances. This would yield more accurate distance estimations individually than using the same regression line for all devices [6]. The idea was to take the average value of these regression lines to estimate distances up to 15 meters without thinking about various antenna specifications.

The known devices were split into three smaller groups to achieve time efficiency. The first group started at a distance of one meter from the sensor and was moved backwards approximately one minute later. When the group had moved 15 meters backwards, the second and third group repeated what group one had completed. During the experiment, start and stop commands were sent at different distances. For each stop command, the raw data was placed in a log file that contained a logical file name, which made the filtering process easier. There was a clear Line of Sight (LOS) between the sensor and the devices to reduce the signal reflection.

The second experiment involved the testing of a software introduced in Section 2.2 that is able to capture Bluetooth Low Energy data frames from Apple devices. The goal was to see if the Apple Bleee software could capture these data packets from distances up to 15 meters. Since this experiment was carried out in the same way as experiment one and because it is different from the other scenarios, this experiment is included in the preliminary scenario.

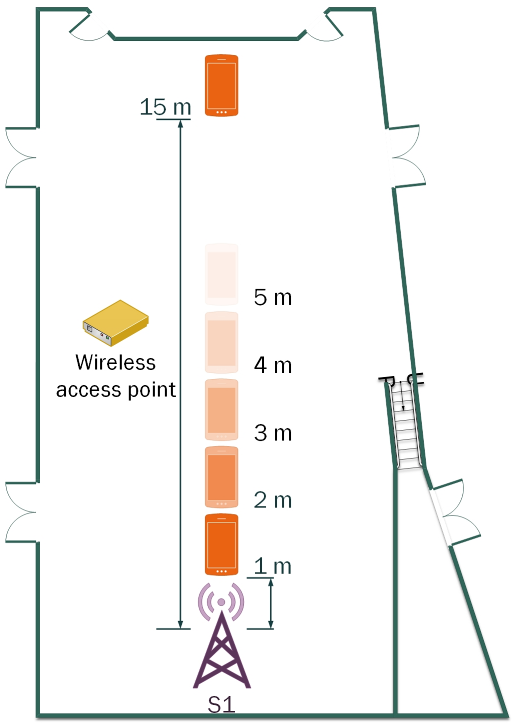


**Figure 4.2:** Illustration of how the preliminary scenario was performed

### Experiment 1 - Preliminary with known devices

The collected data from this experiment consisted of log files from TCPdump, HCI tool and Ubertooth.

### Experiment 2 - Preliminary, experimental with Apple iPhones

In this experiment, the tool called Apple Bleee was tested to see if screen status (idle, lock, home, off) from different Apple products could be captured at different distances. Three iPhone's and one iPad (equipment list in Appendix A.3) were included in this experiment. The experiment was saved as a screen recording from the program's graphical interface. A number in the recordings indicated the distance between the sensor and the device.

### 4.3.2 Scenario 1 - Fixed locations

The first scenario, illustrated in Figure 4.3, show where the ten known devices were located among the fixed positions. Distances from every location to each sensor node was beforehand measured for later verification. The data acquisition started before each device was streaming and before they were connected to a Bluetooth device. The idea was to see if Blue Hydra could reveal information faster than if the acquisition had started after the devices had connected to their Bluetooth device. Because of the large amounts of data that TCPdump generated, the service had to be stopped after 60 seconds. The rest of the services continued for approximately 2 minutes. In this period, approximately 500-700 MB of raw data was collected, including logs from TCPdump, HCI tool and Ubertooth. A closer look at the challenge related to data size indicated that the process of zipping and transferring data back to the controlling node was taking too long when data began to exceed 800 MB. Therefore, in scenario 1-3, the average acquiring time for TCPdump was one minute, while Ubertooth and HCI tool continued for two more minutes.

#### Experiment 3.1, scenario 1, known devices

Each device was placed 30-50 cm above floor level. Device number one was placed in the first location, the second device in location two, etc. An overview of devices can be seen in Appendix A.3.

#### Experiment 6.1, scenario 1, unknown devices

During the experiment, the participants were told to stand upright and hold the device in front of them. The number that was given to the participant in the consent form reflected which position they were placed in. Number one was placed in the first location, etc. During this experiment, one device had a random MAC address. The solution to this has been to change the list of MAC addresses in Matlab to contain this device random MAC address in experiment 6-1 and 6-2 while using its public MAC address in experiment 6-3.

### 4.3.3 Scenario 2 - Fixed locations, different start-up procedure

In the second scenario, the devices were also located at fixed positions. Figure 4.3 can also describe this scenario. Unlike scenario one, the command to acquire data was sent after the devices connected to an external Bluetooth device and began streaming the video. The theory was that Blue Hydra would need more time in this scenario to reveal necessary information about MAC addresses.

#### Experiment 3.2, scenario 2, known devices

The first known device was placed in the first location, etc.

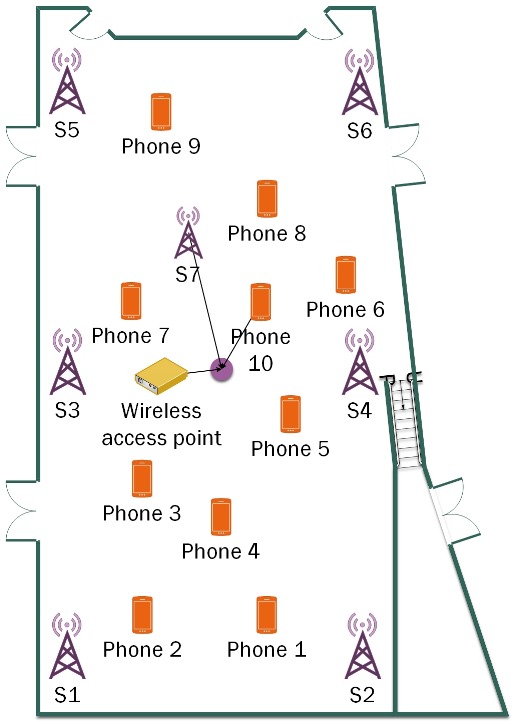


**Figure 4.3:** Illustration of scenario 1-3. Phone 1-10 are indicating the fixed locations

### Experiment 6.2, scenario 2, unknown devices

During the experiment, the participants stood upright while holding their device in front of them. The number that was given to the participant in the consent form reflected which position they were placed in.

### 4.3.4 Scenario 3 - Fixed locations, random Wi-Fi MAC enabled

Wi-Fi MAC randomisation was enabled on devices that supported this function. The start command was sent after the devices had begun streaming video and were playing audio on their connected Bluetooth device. Figure 4.3 can also describe this scenario.

#### Experiment 3.3, scenario 3, known devices

The first device was placed in the first location, etc.

#### Experiment 6.3, scenario 3, unknown devices

During the experiment, the participants stood upright while holding their device in front of them. The number that was given to the participant in the consent form reflected which position they were placed in.

### 4.3.5 Scenario 4 - Fixed pattern, moving as one or in groups

Moving on to the dynamic scenarios. The devices were in this scenario going to move as Figure 4.4 illustrates. The participants were told to pause for 20 seconds when arriving at location 2, 3, 5, 6 and 8, as illustrated in red. When arriving at the purple locations, they should pass by. When the first device began to move against location three, the next device should start. This would eventually follow a system called First-In-First-Out (FIFO).

Data were stored locally on each sensor node from this scenario to save time in further experiments. This also made it possible to acquire data for more than three minutes without waiting for the experiment to complete zipping and transferring when the stop command was sent.

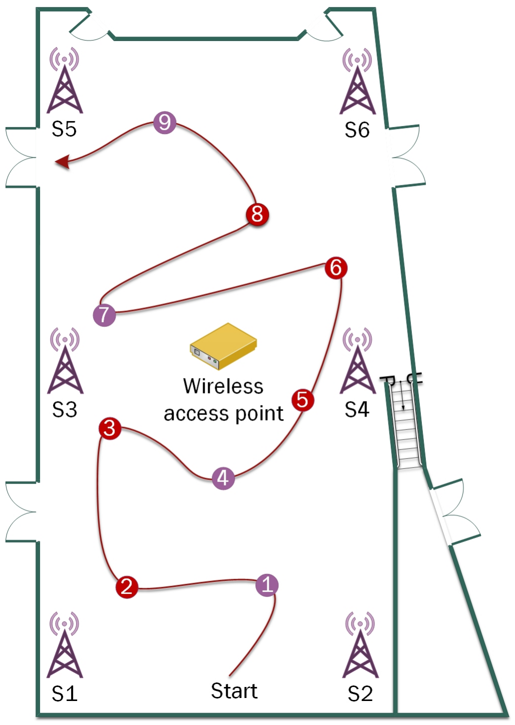


**Figure 4.4:** Illustration of scenario 4-5 with the pre-defined route

#### Experiment 4, scenario 4, known devices, performed with normal speed

The devices rested for twenty seconds on location 2, 3, 5, 6 and 8. One person held several devices during the experiment for time efficiency.

#### Experiment 7.1, scenario 4, unknown devices, performed at a lower speed

The participants were instructed to line up in ascending order with their number as a basis. Further, they were told to follow a prepared trail that took them from one side of the room to the other. During the experiment, it was approximately 20 seconds between the participants.

#### Experiment 7.2, scenario 4, unknown devices, performed at a higher speed

Same as in experiment 7.1, but with higher speed. Now, only stopping for three seconds in the red locations (Figure 4.4). This experiment was performed to see if the results changed when moving faster than normal. The data logs in this experiment had a smaller size as the participants spent less time on the track.

### 4.3.6 Scenario 5 - Fixed pattern, move as one group

The purpose of scenario five was to track devices when they moved close to each other through the pre-defined route in Figure 4.4. For practical reasons and to comply with distance rules related to covid-19, only the group of known devices were attending this scenario.

#### Experiment 5, scenario 5, known devices

The person holding these devices walked normally and stopped for 20 seconds at locations 2, 3, 5, 6 and 8. As Figure 4.5 shows, the devices kept almost the same angle during the experiment with some distance between each other. All Bluetooth devices were placed in the compartment at the back of the cardboard box. An early observation in this experiment was that data related to one device (relatively new model) only collected a few probe requests instead of metadata from Transmission Control Protocol (TCP) packets.

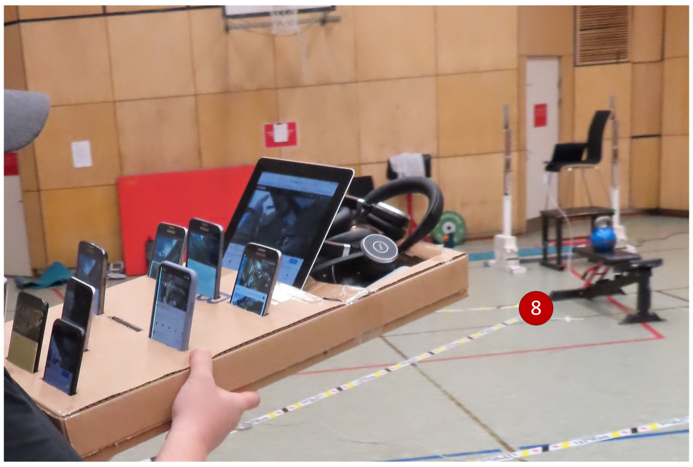


**Figure 4.5:** Picture from experiment five showing how it was performed

## 4.4 Experiment setup flowchart

An overview of the experiment setup chapter is presented as a flowchart in Figure 4.6. The first step involved the preparation of various scenarios and experiments. These scenarios and experiments were further described in detail in the collection plan, which served as a manual during the data collection. In front of the data collection, a consent form was created to take care of GDPR concerns. The consent was reviewed and approved by legal professionals before the participants were invited and willingly signed the consent form. The next phase consisted of preparations that needed to be completed to perform the data collection, such as network stability, video recording and battery charging. Before collecting the data, we ensured ground truth by measuring the distances between the fixed locations and the sensors.

The data collection phase took place over three days with constant work. During day one, experiment one and two, consisting of preliminary work, were completed. The next day included static and dynamic scenarios consisting of experiment three through five with known devices. Lastly, day three consisted of static and dynamic scenarios with ten participants that performed experiment six through seven. These participants had beforehand signed the consent form. After the collection phase was completed, the acquired data were gathered, verified and placed on a specific encrypted laptop with additional backup to an external hard drive.

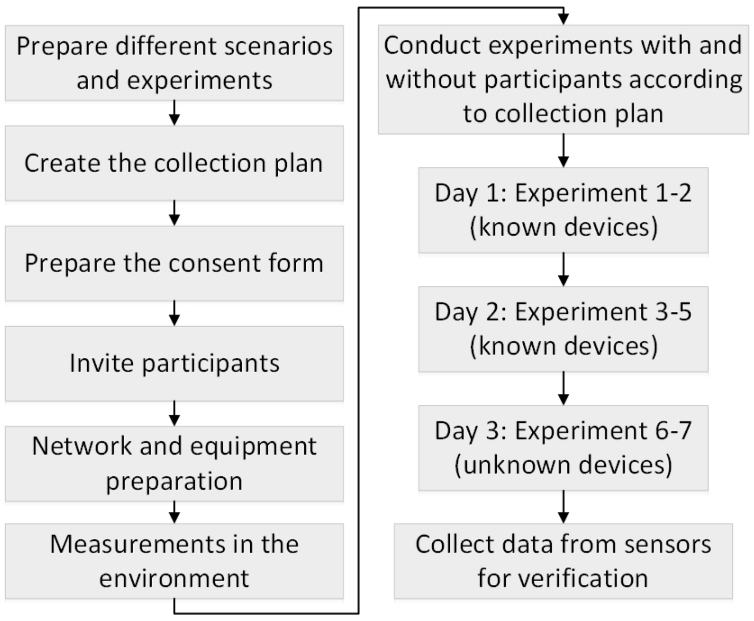

Prepare different scenarios and experiments
Create the collection plan
Prepare the consent form
Invite participants
Network and equipment preparation
Measurements in the environment
Conduct experiments with and without participants according to collection plan
Day 1: Experiment 1-2 (known devices)
Day 2: Experiment 3-5 (known devices)
Day 3: Experiment 6-7 (unknown devices)
Collect data from sensors for verification

**Figure 4.6:** Different phases of the experiment setup illustrated as a flowchart

# Chapter 5

# Pre-processing and data analysis

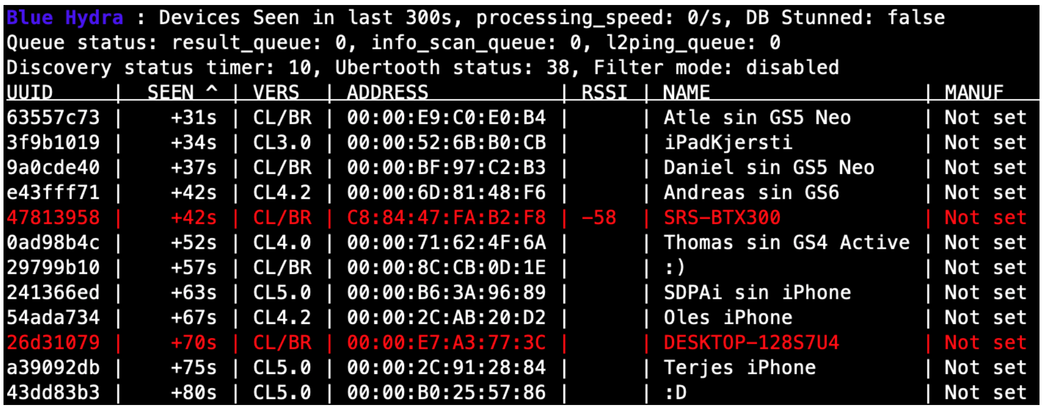
This chapter dives into the collected data to analyse how the data was filtered and cleaned and how the data was further used in signal correlation and geolocation methods. This chapter will also look into Blue Hydra and Apple Bleee analysis, describe the different data types within the data logs, elaborate on how statistical averaging was performed and analyse data from different experiments. The chapter will provide the reader with a detailed perspective of how the data was processed and utilised.

## 5.1 Blue Hydra and Apple Bleee analyses

### 5.1.1 Blue Hydra analysis

In scenario one and two, Blue Hydra managed to reveal the Lower Address Part (LAP) (SAP minus UAP) and some of the public MAC addresses among the attending devices. In addition, Blue Hydra managed to reveal the device names, which essentially would hint as to which device belongs to whom. Among the group of unknown devices, there were ten unique device names. Further on, the Lower Address Part were used in Matlab to whitelist interesting devices listed in the Ubertooth logs. Figure 5.1 shows a screenshot taken in the graphical user interface of Blue Hydra when the known devices were streaming a YouTube video while using a Bluetooth device as the audio source.

An interesting finding while using Blue Hydra show that devices searching for their Bluetooth device inside the Bluetooth menu increases the probability for Blue Hydra to reveal their full public MAC address. However, this does not seem valid for fast Bluetooth re-connection. Although a complete MAC address or SAP has higher credibility than the LAP address, the latter will provide the law enforcement with enough uniqueness to distinguish between devices. Scenario one and scenario two show that Blue Hydra is not finding the LAP address faster if a phone connects to their Bluetooth device in a normal way, while Blue Hydra is sniffing. The exception is if the phone spends several seconds inside the Bluetooth menu or begins a new pairing.



```
Blue Hydra : Devices Seen in last 300s, processing_speed: 0/s, DB Stunned: false
Queue status: result_queue: 0, info_scan_queue: 0, l2ping_queue: 0
Discovery status timer: 10, Ubertooth status: 38, Filter mode: disabled
UUID      |  SEEN ^ | VERS  | ADDRESS           | RSSI | NAME                  | MANUF
63557c73 |    +31s | CL/BR | 00:00:E9:C0:E0:B4 |      | Atle sin GS5 Neo      | Not set
3f9b1019 |    +34s | CL3.0 | 00:00:52:6B:B0:CB |      | iPadKjersti           | Not set
9a0cde40 |    +37s | CL/BR | 00:00:BF:97:C2:B3 |      | Daniel sin GS5 Neo    | Not set
e43fff71 |    +42s | CL4.2 | 00:00:6D:81:48:F6 |      | Andreas sin GS6       | Not set
47813958 |    +42s | CL/BR | C8:84:47:FA:B2:F8 | -58  | SRS-BTX300            | Not set
0ad98b4c |    +52s | CL4.0 | 00:00:71:62:4F:6A |      | Thomas sin GS4 Active | Not set
29799b10 |    +57s | CL/BR | 00:00:8C:CB:0D:1E |      | :)                    | Not set
241366ed |    +63s | CL5.0 | 00:00:B6:3A:96:89 |      | SDPAi sin iPhone      | Not set
54ada734 |    +67s | CL4.2 | 00:00:2C:AB:20:D2 |      | Oles iPhone           | Not set
26d31079 |    +70s | CL/BR | 00:00:E7:A3:77:3C |      | DESKTOP-128S7U4       | Not set
a39092db |    +75s | CL5.0 | 00:00:2C:91:28:84 |      | Terjes iPhone         | Not set
43dd83b3 |    +80s | CL5.0 | 00:00:B0:25:57:86 |      | :D                    | Not set
```

**Figure 5.1:** Blue Hydra revealing SAP addresses (UAP + LAP) of known devices. The column describing the type is removed for best scaling

### 5.1.2 Apple Bleee analysis

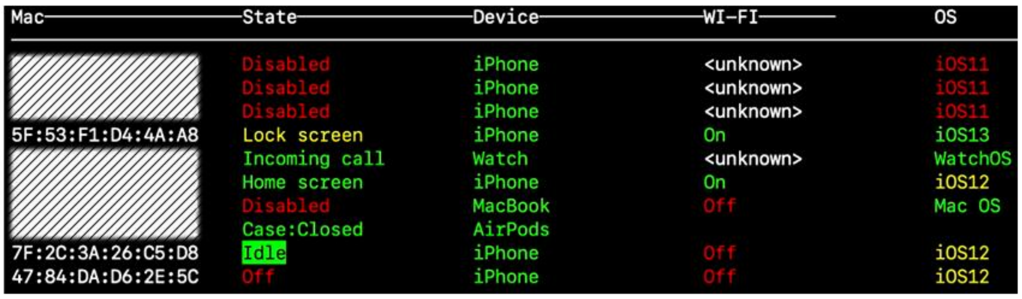
Experiment two in this master's thesis includes a test of Apple Bleee with four Apple products at different distances. This was mainly to see if the sensor would pick up different screen status at 15 meters. The experiment may seem tedious, but from a mobile forensic perspective, it is essential whether the user has entered the screen lock or not after the last restart. The fact is that an After First Unlock (AFU)[1] extraction will contain encrypted information from third-party applications. In contrast to AFU, a Before First Unlock (BFU)[1] extraction only contains general information from the device. Utilising the Apple Bleee software combined with covert police work can give basic information on whether the device is in use or at rest, affecting when a person should be arrested.

Figure 5.2 shows three iPhone's in the graphical user interface of Apple Bleee at 15 meters with forensically important screen status and useful information about the iOS version. The latter will indicate whether the forensic tools which the law enforcement use is able to support the device or not. The last Apple product (iPad) is not listed as it does not support Bluetooth Low Energy. The first column containing the random MAC address changed rapidly during the experiment, making it useless for tracking.

Different screen status that is tested:

- Idle: screen off
- Lock screen: lock screen
- Home screen: user entered code, on home screen
- Off: lock screen + idle

[1]Explained in the list of glossary



```
Mac                  State                Device               WI-FI                OS

[redacted]           Disabled             iPhone               <unknown>            iOS11
                     Disabled             iPhone               <unknown>            iOS11
                     Disabled             iPhone               <unknown>            iOS11
5F:53:F1:D4:4A:A8    Lock screen          iPhone               On                   iOS13
[redacted]           Incoming call        Watch                <unknown>            WatchOS
                     Home screen          iPhone               On                   iOS12
                     Disabled             MacBook              Off                  Mac OS
                     Case:Closed          AirPods
7F:2C:3A:26:C5:D8    Idle                 iPhone               Off                  iOS12
47:84:DA:D6:2E:5C    Off                  iPhone               Off                  iOS12
```

**Figure 5.2:** Screen status from iPhones obtained with Apple Bleee at 15 meters

## 5.2 Data types and metadata in the collected data

A data packet consists of information from several data types and is found in both the log files captured by TCPdump and Ubertooth. The data packets included these types of data: source, signal strength, channel, and name. Table 5.1 is describing the the data packets more in detail. To make the data more searchable and recognisable, we added metadata with the Ansible script that contained the following data types: id, date, type, experiment, sensor, filename and file path. In Table 5.2 a brief description of these data types is given. The metadata and data types mentioned here is what we found as useful information that could be captured passively from Bluetooth and Wi-Fi.

| **Data type** | **Description of captured data types** |
|---|---|
| Time | Each data packet contained a timestamp. For the Ubertooth tool, these timestamps were converted from Unix epoch time to UTC+0 (down to seconds). The TCPdump tool already had this converted correctly (down to milliseconds) |
| Source | This field represents the Wi-Fi MAC address captured by TCPdump or the Bluetooth Lower Address Part (LAP) captured with Ubertooth |
| Signal strength | The received signal strength at the antenna measured in dBm. This value is described more in Section 2.1.1 |
| Channel | Indicates which channel the data packet was sent on. Most of the data packets collected by TCPdump had channel 11, while data packets captured with Ubertooth showed channels between 0-78 |
| Name | Data field in packets captured with TCPdump that often contain the name of the manufacturer such as: SamsungE, Apple, OnePlusT, Google |
| Packet number | In TCPdump, each data packet had a packet number that was imported into SQL. This may be used in further work |

**Table 5.1:** Data types captured by TCPdump and Ubertooth

| Data type | Description of metadata added |
|---|---|
| Id | Each data packet was given a unique number during the import to SQL |
| Date | Date information in each data packet and in the title of each log file |
| Type | Log type, such as: Ubertooth, TCPdump |
| Experiment | Name of the experiment that the data packet originated from |
| Sensor | A number associated with the sensor that captured the data packet |
| Filename | A name that was given to each log file to make it more searchable |
| File path | Included as a data type to be sure where the data comes from |

**Table 5.2:** Data types that were added to each data packet as metadata

## 5.3 Data filtering and cleaning methods in detail

In Section 3.4, the methods for filtering and cleaning data was presented. Some of those methods will be described more in detail, such as (i) unwanted observations, (ii) distance-based filtering, (iii) time compression and (iv) missing data.

### Unwanted observations

In order to avoid importing irrelevant data to the SQL database, a method called whitelisting[2] was applied as a filtering mechanism in Wireshark to those devices that were attending the experiments. One of these filtering commands, presented in Code Listing 5.1, was used to filter and verify pcap-files generated by TCPdump. Also, Wi-Fi packets with frequency channel different from 11 and received signal strength represented as null values were removed. These unwanted observations and the whitelisting reduced a vast amount of irrelevant data. In contrast to Wi-Fi packets, the raw data files related to Ubertooth (txt-files) were not cleaned before importing to SQL because of the log simplicity and insignificant data size. However, the Bluetooth data captured by Ubertooth were opened in Wireshark for data verification. Code Listing 5.2 shows the whitelisting filter that was used to verify that each Bluetooth device was included in the Ubertooth logs.

**Code listing 5.1:** Wi-Fi filtering in Wireshark

```
% Whitelisting: Filter on transmitter address (TA) and source address (SA)
% Signal strength: less or equal to 0
% Frequency: Show Wi-Fi packets sent on frequency channel 11
((wlan.ta == 04:15:52:6B:B0:CA) || (wlan.sa == 04:15:52:6B:B0:CA)) &&
(wlan_radio.signal_dbm <= 0) && (wlan_radio.channel == 11)
```

**Code listing 5.2:** Bluetooth filtering in Wireshark

```
% Whitelisting: Only show these LAP addresses
(btbredr_rf.lower_address_part == 0x006BB0CB) % or (btbredr... == next LAP address)
```

[2]Explained in the list of glossary

Another filtering method used to remove unwanted observations was time filtering. This filter was applied in those dynamic scenarios where the devices were following a pre-defined route. The video recordings gave the actual time and location, and this information was used in Matlab to keep those data packets captured between start and finish. The data packets outside these timestamps were excluded.

### Distance-based filtering

Section 3.4 gave a brief explanation of the distance-based filtering method. This section will describe in detail how the thresholds for Wi-Fi and Bluetooth were calculated. First, in order to find the lower threshold for Wi-Fi signals, the free-space path loss Equation 5.1 was used. The longest distance within the room was measured to $d = 23.75\,meters$, while channel 11 was set as the frequency ($f = 2462\,MHz$). The speed of light is still constant ($c = 300 \cdot 10^6 m/s$). Taking into account that a human body attenuates signals approximately by 3 dBm (halves the signal), this value was added in the link budget as path loss [46]. In addition, a typical transmitted Wi-Fi signal has its output power between 15-20 dBm (30-100 mW) [17]. This master's thesis chose the transmit power equal to $20\,dBm$, which is the maximum transmit power of Wi-Fi. This fits better with this data set than if the value were, e.g. $15\,dBm$. However, in contrast, to probe requests that are transmitted louder than data packets acquired in this thesis, newer devices can transmit data packets by adjusting the output power dynamically [17][47]. This includes Wi-Fi devices with newer standards, e.g. radio measurement extensions (802.11k/r/v) and Multiple-Input Multiple-Output (MIMO)[3] where output power potentially is based on the distance from the device to the wireless access point [47]. Since this thesis is not accounting for dynamic output power, this could be a limitation. However, it may still be usable for law enforcement. Equation 5.2 show the power link budget for the lower threshold, including a 10 % uncertainty, resulting in a limit of 56 dBm. Thus, several low values that could be associated with multipath are filtered out.

When it comes to the upper threshold, this was set to filter out received signals stronger than -15 dBm. The threshold was chosen because the number of received signals in the collected data outside this limit was rare and appeared as misleading. Also, the methods seem to be more precise when setting this upper threshold. Based on the facts mentioned above, the Wi-Fi signal thresholds are given in 5.3.

$$FSPL(dB) = 10log\left(\left(\frac{4\pi \cdot 23.75 \cdot 2462}{300 \cdot 10^6}\right)^2\right) = 67,78\,dBm \tag{5.1}$$

$$\Rightarrow (67,78\,dBm + 3\,dBm - 20\,dBm) + 10\,\% \approx 56\,dBm \tag{5.2}$$

$$\Rightarrow -56\,dBm \leq \text{ Wi-Fi signal} \leq -15\,dBm \tag{5.3}$$

[3]Explained in the list of glossary

Changing the focus over to Bluetooth, the frequency is set to $f = 2441\, MHz$, which is the average frequency of all 79 Bluetooth Classic channels. The Bluetooth signal path loss is also based on the distance equal to $d = 23.75\, meters$ and is calculated with Equation 5.4. When we adjusted the link budget for body attenuation (3 dBm) and assumed that the devices transmitted with a maximum output power (4 dBm / 2.5 mW, class two devices), the lower Bluetooth signal threshold was set to $73\, dBm$ including a 10 % uncertainty. The upper threshold is also in this technology set to -15 dBm. Expression 5.5 show the lower and upper Bluetooth signal thresholds which are used in this master's thesis.

$$FSPL(dB) = 10log\left(\left(\frac{4\pi \cdot 23.75 \cdot 2441}{300 \cdot 10^6}\right)^2\right) = 67,71\, dBm \tag{5.4}$$

$$\Rightarrow (67,71\, dBm + 3\, dBm - 4\, dBm) + 10\,\% \approx 73\, dBm$$

$$\Rightarrow -73\, dBm \leq \text{ Bluetooth signal} \leq -15\, dBm \tag{5.5}$$

Code Listing 5.3 and 5.4 are showing how the distance-based filter was performed in SQL for Wi-Fi and Bluetooth.

**Code listing 5.3:** Distance-based filtering in SQL for Wi-Fi

```sql
insert into tcpdump_whitelisted_distance
SELECT *
FROM tcpdump_updated
WHERE (Signal_strength_dBm_2 >=-56 AND Signal_strength_dBm_2 <=-15)
```

**Code listing 5.4:** Distance-based filtering in SQL for Bluetooth

```sql
insert into ubertooth_whitelisted_distance
SELECT *
FROM ubertooth_whitelisted
WHERE (Signal_strength_dBm >=-73 AND Signal_strength_dBm <=-15)
```

### Time compression in dynamic scenarios

In the acquired data, huge amounts of data packets, especially from TCPdump, were received within the same second. Hence, to prepare the data for the dynamic scenarios, it was necessary to compress these data packets into the same second. One second should be associated with one signal value. The solution was to group the data packets by time, sensor and source, while the average signal strength was calculated from all the packets associated with this group. The result, which significantly reduced the number of data rows, made the data more convenient in Matlab. For the static scenarios, time compression was unnecessary as all devices were located in the same place during the experiment. In the static scenarios, the average signal strength of all received data packets was calculated. More about that in Section 5.4.

Time compression, as described above, generated two additional tables in SQL. These tables were only used in the geolocation scenarios when the devices were moving, described in Section 4.3.5 and 4.3.6.

- tcpdump_whitelisted_combined_seconds_distance
- ubertooth_whitelisted_combined_seconds_distance

Code Listing 5.5 shows how time compression was applied to the Wi-Fi data packets captured by TCPdump. The same procedure was performed on the Bluetooth data packets captured by Ubertooth.

**Code listing 5.5:** Time compression filter applied in SQL

```sql
insert into tcpdump_whitelisted_combined_seconds_distance
SELECT *,
MAX(Signal_strength_dBm_2) AS "Signal_strength_dBm_2_high",
AVG(Signal_strength_dBm_2) AS "Signal_strength_dBm_2_Average_Limit56",
count(*) AS "Combined",
strftime('%H:%M:%S', time) as "Time_2"
FROM tcpdump_updated
group by strftime('%H:%M:%S', time), Sensor, substr(Source, -8)
```

### Missing data and error matrix

As described in Section 3.4, mechanisms were implemented into the Matlab script to discover missing data packets. These mechanisms discovered among the Bluetooth data that packets associated with the known phone 9 were not captured during experiment 3-2, 3-3, 4 and 5. This was probably because the device was not playing the YouTube video during these experiments or because the Bluetooth device was disconnected. When looking at the Wi-Fi data, it was discovered that the known phone 1 was missing in experiment 3-3 and 5. Only a few packets were captured from Phone 2, so that data was discarded. This means that the results will have a deviation with these devices in mind.

We realised that capturing data packets with the HCI tool and L2ping were not as robust as expected in early tests. Several attempts to improve the Ansible script did not solve the problem: to use L2ping in parallel with the HCI tool to capture data packets from several devices simultaneously. Of those few data packets that were captured with the HCI tool in parallel with L2ping, these were only associated with some devices and were only captured at some sensors. Because of this, these packets were discarded and is not analysed further in this master's thesis. However, the data packets captured with Ubertooth and TCPdump included nearly all devices and was well distributed between the sensors. Table 5.3 gives an overview of the missing data packets from the attending devices in different experiments between different capturing tools.

### The impact of filtering and cleaning methods

To give an overview of the impact of the above-mentioned filtering methods, Table 5.4 and Table 5.5 show the amount of Wi-Fi and Bluetooth data packets captured

| | **Experiments** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 3-1 | 3-2 | 3-3 | 4 | 5 | 6-1, 6-2, 6-3 | 7-1, 7-2 |
| TCPdump | All | All | All | 8/10 | All | 9/10 | All | All |
| Ubertooth | All | All | 9/10 | 9/10 | 9/10 | 9/10 | All | All |
| HCI tool | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | - | - |
| L2ping | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | - | - |

**Table 5.3:** Error matrix. Nearly all devices are present in the captured data related to TCPdump and Ubertooth

in each experiment. The tables also show the impact of packet reduction when applying to each filtering method.

| | | Wi-Fi data packets *after* filtering | | |
|---|---|---|---|---|
| **EXP** | **Wi-Fi data packets** | **Unwanted observations** | **Distance based** | **Time compression** |
| 1 | 1 226 239 | 784 226 | 736 401 | - |
| 3-1 | 2 008 234 | 360 905 | 262 550 | - |
| 3-2 | 1 010 121 | 178 697 | 151 255 | - |
| 3-3 | 454 585 | 72 487 | 56 979 | - |
| 4 | 2 934 196 | 462 214 | 378 636 | 6 785 |
| 5 | 2 537 181 | 386 976 | 322 136 | 8 466 |
| 6-1 | 795 604 | 163 924 | 144 837 | - |
| 6-2 | 432 437 | 77 077 | 69 128 | - |
| 6-3 | 862 703 | 173 562 | 157 934 | - |
| 7-1 | 4 215 968 | 691 882 | 591 664 | 20 527 |
| 7-2 | 1 312 388 | 236 293 | 206 123 | 4 496 |
| Total | 17 789 656 | 3 588 243 | 3 077 643 | 40 274 |

**Table 5.4:** The total amount of Wi-Fi data packets captured by TCPdump in different experiments and the remaining data packets after different filtering methods

| | | Bluetooth data packets *after* filtering | | |
|---|---|---|---|---|
| **EXP** | **Bluetooth data packets** | **Unwanted observations** | **Distance based** | **Time compression** |
| 1 | 45 136 | 29 149 | 29 023 | - |
| 3-1 | 24 856 | 14 418 | 12 865 | - |
| 3-2 | 15 500 | 9 843 | 9 338 | - |
| 3-3 | 15 531 | 11 265 | 10 887 | - |
| 4 | 100 386 | 76 178 | 62 463 | 11 095 |
| 5 | 29 553 | 21 647 | 20 929 | 6 076 |
| 6-1 | 32 143 | 17 167 | 16 356 | - |
| 6-2 | 22 099 | 13 608 | 13 033 | - |
| 6-3 | 45 777 | 22 023 | 20 827 | - |
| 7-1 | 85 465 | 51 263 | 47 316 | 15 907 |
| 7-2 | 29 115 | 16 976 | 15 843 | 4 389 |
| Total | 445 561 | 283 537 | 258 880 | 37 467 |

**Table 5.5:** The total amount of Bluetooth data packets captured by Ubertooth in different experiments and the remaining data packets after different filtering methods

## 5.4 Statistical averaging and general calibration

### Statistical averaging in static experiments

Section 5.3 describes how time compression was calculated in the dynamic experiments. This section describes the adjustments and calculations that were performed on the received signals in the static experiments. The goal was to associate one value of signal strength per device per sensor per experiment. In order to achieve this, statistical averaging was necessary. Both the mean and median value was tested in several experiments to find which that fitted best. By analysing the data, we found that the overall received signals varied greatly (e.g. strongest Bluetooth signal was measured at -29 dBm, while the weakest was measured to be -73 dBm). Since the median value is better for data with similar values, the mean averaging method was chosen. This means that the sum of received signal values (associated with the same device, sensor and experiment) was divided by the number of data packets. This approach was common for all the static experiments, including 1, 3-1, 3-2, 3-3, 6-1, 6-2 and 6-3.

### Calibrating sensors and individual output power

Research performed by Kolberg [1] show that individual calibration of sensors and output power improved the estimated distance between the sensor and the device. Such calibration was also performed manually in this thesis, where we used a reference device from experiment 3-1. This device was located in the middle of the room, where the actual distances were known. Code Listing 5.6 shows the specific values that calibrated each sensor. By doing this, the estimated distance become more accurate compared to earlier estimations. However, the calibration was technology-dependent, which generated some values associated with the Wi-Fi data and some for the Bluetooth data.

**Code listing 5.6:** Implementing the calibrating of sensors in Matlab

```
% Based on static exp. 3-1 with known Phone #10 located in the middle of the room
    % Wi-Fi specific
% Constant to compensate for PR in each sensor. First value relates to sensor 1, etc
PR_constant_wifi = [5;5.2499;6.1701;5.1694;-0.2861;5.1];
    % Bluetooth specific
% Constant to compensate for PR in each sensor. First value relates to sensor 1, etc
PR_constant_bluetooth = [-1.574;-2.2037;2;-0.605;-2.173;-1.9316];

% Add mean value of the current data to the specific sensor constant
PR_powerreceiver = PR_constant_wifi(s) + mean(current_data{:,4});
```

Bluetooth and Wi-Fi specification show that devices usually send with power equal to 20 dBm (Wi-Fi) and 4 dBm (Bluetooth) [17]. During the calculation of estimated path loss, these values were first used. Later, it occurred that the output power could be device individual. We divided the standard output power into a general value (PT_Wi-Fi = 15 dBm and PT_Bluetooth = 8 dBm) and a device-individual constant to mitigate the unknown output power. Because this calibration was time-consuming and would not be realistic to calibrate unknown devices, only the known devices were individually calibrated. The unknown devices used the general output value. Code Listing 5.7 gives an overview of the individual values that were manually tested to find the value which gave the most accurate distance. Both the calibrated PR and PT values were used directly into the free-space path loss formula 3.1.

**Code listing 5.7:** Implementing the calibrating of output power in Matlab

```
% Based on static EXP 3-1 with known Phone #10 located in the middle of the room
    % Wi-Fi specific
% Power Transmit Wi-fi - Reference number (Normal output should be 20 dBm)
PT_powertransmit_wifi_known = 15;
% Constant to compensate for PT, number 10 is zero because it’s the reference device
PT_constant_wifi = [2.2;-4;2.4;1;3;0;-1;2.5;0;0];
    % Bluetooth specific
% Power Transmit Bluetooth - Reference number (Normal output should be 4 dBm)
PT_powertransmit_bluetooth_known = 8;
% Constant to compensate for PT, number 10 is zero because it’s the reference device
PT_constant_bluetooth = [4;-1.15;3;1;2;-3;5;-3;-0.3;0];

% Add the general output value to the device specific constant
PT_powertransmit_current = PT_powertransmit_wifi_known + PT_constant_wifi(m);
```

## 5.5 Signal correlation analysis

In this section, the data from a static experiment have been analysed to give an overview of how Bluetooth and Wi-Fi signals were linked. The goal was to determine if the partially unique Bluetooth LAP address and the Wi-Fi MAC address were coming from the same device. Data from TCPdump and Ubertooth captured in the static experiments (3-1, 3-2, 3-3, 6-1, 6-2 and 6-3) have been tested with the linking methods mentioned in Section 3.8.

Figure 5.3 show data from four devices that were attending experiment 3-1. On the x-axis, the devices are grouped to the sensor which received the signals. Bluetooth and Wi-Fi signals coming from the same device are presented side by side in each group, with their absolute average signal strength on the y-axis.

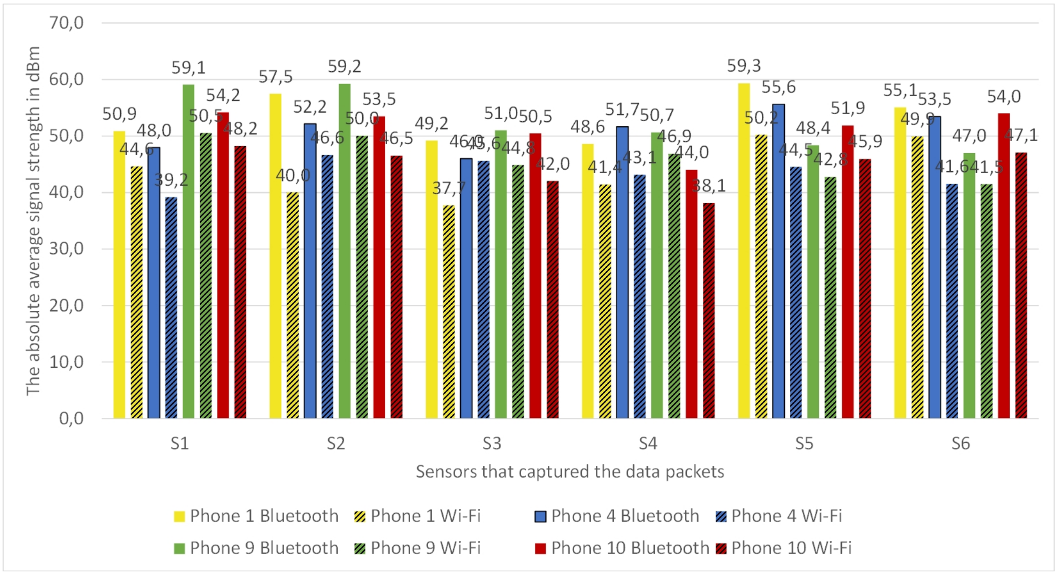


**Figure 5.3:** Absolute average signal strength from four devices in experiment 3-1. Each signal strength is rounded to the nearest decimal

By converting the signal values into distances with Equation 3.4, it was possible to compare the distances with ground truth. This conversion is performed in Figure 5.4. According to the numbers, sensor 1 and 2 have received weaker signal values from phone 9 than phone 1 and phone 4. On the opposite side of the room, sensor 5 and 6 have received the strongest signal from phone 9. This agrees quite well with the actual distances, as device 9 is 3.43 meters away from sensor 5 and 8.25 meters away from sensor 6. Although the information deviates from the actual distances, it may give us a pattern to link Bluetooth and Wi-Fi signals coming from the same device.

There will always be uncertainty associated with the measured signal values that will affect the accuracy when the signals are linked. One uncertainty is related to the dynamic output power that the device with newer hardware could change

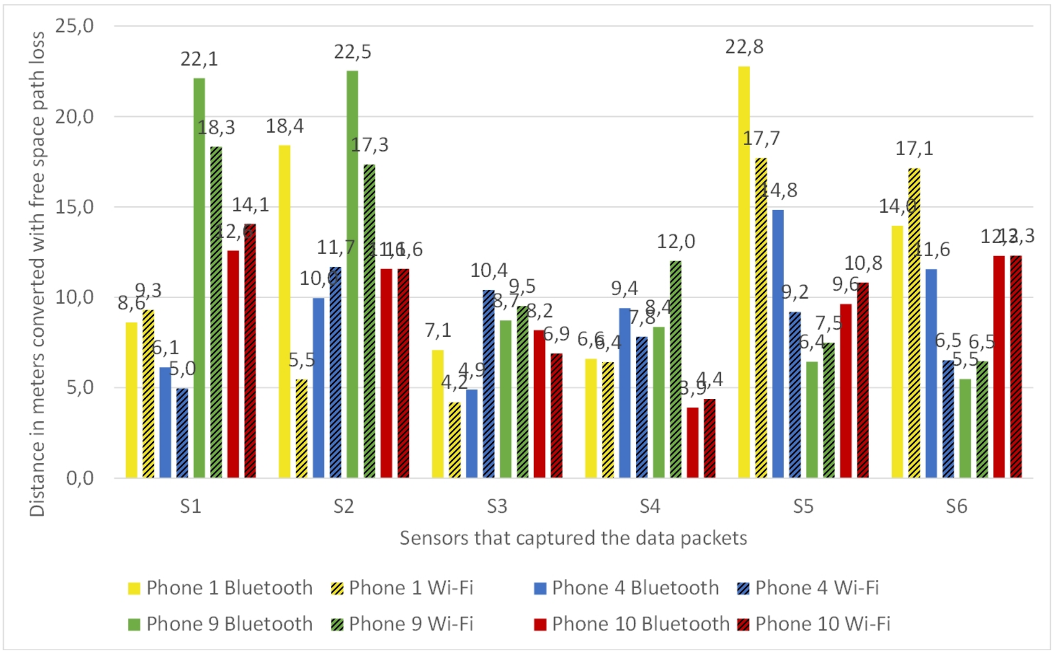


**Figure 5.4:** Received signals from Figure 5.3 converted into distances. Each distance is rounded to the nearest decimal

depending on the distance. In this thesis, we have accounted for a constant output power (both Bluetooth and Wi-Fi). Some uncertainty is also associated with the data packets as the data packets are captured passively and received in an unordered sequence. Because of this, data packets that have suffered from multipath may arrive later and inaccurately affect the result. With only four devices in the room, it would be easier to link Bluetooth and Wi-Fi signals originating from the same device. However, as we will see, this is more challenging with several devices in close vicinity. To present the accuracy of the linking algorithms, the section about these results will use a top k-nearest neighbours approach. The approach involves searching the entire data set for the k number of most similar MAC addresses or neighbours showing the closest values. As the euclidean distance between the vectors (associated with the MAC addresses) is presented in ascending order (downwards), the MAC addresses that are closest linked (lowest value) came at the top. That way, it is possible to see if the MAC addresses are linked on the top 1, top 3, top 5, or top k devices.

Frequency analysis from the signal correlation in experiment 3-1 and 6-1 shows that some devices were easier to link correctly in the top 1. These devices are coloured green and listed in Table 5.6. Note that devices with older Android OS top the list in experiment 3-1, while devices with relatively new iOS top the list in experiment 6-1. The remaining devices which were found in the top 3, top 5 and top 10 are also listed in Table 5.6 but not coloured. Results related to signal correlation are presented in the next chapter in Section 6.1.

| **3-1 (known devices)** | | **6-1 (unknown devices)** | |
|---|---|---|---|
| Samsung S5 Neo | Android 6.0.1 | Google Pixel 4 | Android 11 |
| Samsung S5 Neo | Android 6.0.1 | iPhone SE | iOS 13.6.1 |
| Samsung S6 | Android 7.0 | iPhone XR | iOS 13.7 |
| iPhone XS | iOS 13.6.1 | iPhone 8 | iOS 14.1 |
| Samsung S20 | Android 10 | Samsung S20-5G | Android 10 |
| Samsung S8 | Android 9 | iPhone 11 Pro | iOS 14 |
| iPhone SE | iOS 13.5.1 | Samsung S10e | Android 10 |
| iPhone 6 | iOS 12.4.8 | OnePlus 8 Pro | Android 10 |
| iPad 2 | iOS 9.3.5 | iPhone XR | iOS 14.0.1 |
| Samsung S4 Active | Android 5.0.1 | iPhone XR | iOS 14.0.1 |

**Table 5.6:** Devices that occurred most frequently among the linking methods in top 1

## 5.6 Geolocation analysis using signal strength

Triangulation and trilateration are methods performed in both the static experiments (3-1, 6-1) and the dynamic experiments (4, 5, 7-1, 7-2) to estimate where devices are located based on received signal strength. These two methods share some steps that will be described further in this section. This section will also include an example of the data from experiment 3-1, where a device's location will be estimated with geolocation methods to check for inaccuracies. The example is illustrated using triangulation and trilateration that also show the accuracy between Bluetooth and Wi-Fi data.

In the static experiments, where the devices were at rest, the mean value of all received signals associated with the same device, sensor and experiment were calculated. This is already elaborated in Section 5.4. In contrast to the static experiments, the dynamic experiments applied time compression and time filtering described in Section 5.3. Even though time compression reduced huge amounts of data packets into the same second and time filtering excluded irrelevant packets, the data from TCPdump and Ubertooth still included tens of data rows. It eventually became difficult to illustrate and compare the results without further action. A solution performed in this thesis is by scattering every other $x$ estimation to only include around ten estimations. For example, if triangulation estimates around 60 locations based on Bluetooth data, the plot only shows every other five estimations (12/60). This provided better space in the plotted coordinate system and made it easier to present both Bluetooth and Wi-Fi data using triangulation and trilateration.

#### Triangulation analysis using signal strength from Wi-Fi and Bluetooth data

To better understand how triangulation works, data from experiment 3-1 are used to estimate the location of phone 10. Analysis of the estimated location is illustrated in Figure 5.5. In this estimation, 6 out of 16 sensor pairs were used to calculate the mean location. These six estimations are shown with a blue (Bluetooth) or yellow (Wi-Fi) triangle. The two enlarged triangles in blue and yellow represent the mean value of these smaller triangles. The remaining ten triangles were ignored because they either were outside the area or did not comply with the policy described in Section 3.7.2. By taking the mean value of the smaller triangles, the estimation seems to be more accurate. Wi-Fi packets captured with TCPdump seem to be more accurate than the Bluetooth packets captured with Ubertooth. Both estimations are slightly (1.2-2.2 meters) further to the left than the actual location, illustrated with the green circle. A closer look at the estimations from all devices attending experiment 3-1 shows that the estimation of phone 10 is one of the top three.

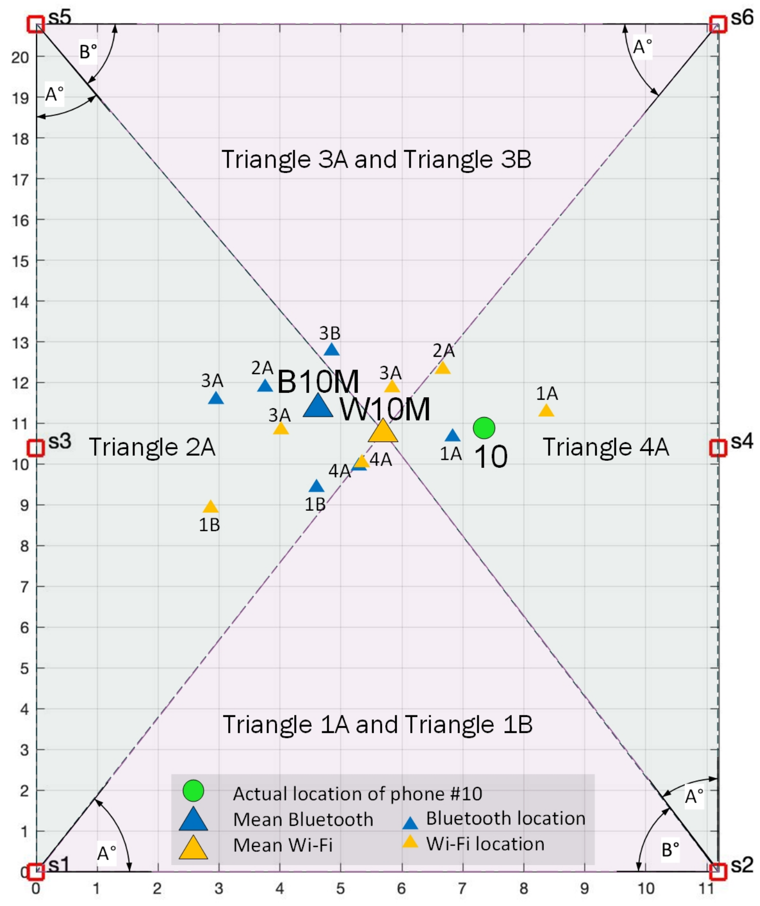


**Figure 5.5:** Triangulation - Estimated location of phone 10 based on different pairs of sensors and mean values from Wi-Fi and Bluetooth data

**Trilateration analysis using signal strength from Wi-Fi and Bluetooth data**

The same data from above is also analysed with trilateration to estimate the location of phone 10. This estimation is illustrated in Figure 5.6 with blue and yellow circles. The smaller circles are the estimates from different compositions of sensors, while the enlarged circles are the mean value of those smaller ones. In this case, the data included enough data packets to estimate the location from four sensor compositions. However, the overall data shows that other devices' estimated locations only consist of two out of four sensor compositions. This will affect the accuracy of the estimations since Figure 5.6 shows that the mean value of several sensor compositions increases accuracy. Even though the mean Wi-Fi estimation is slightly better than the mean Bluetooth estimation, the locations are 1.2-2.2 meters away from the actual position. All results related to experiment 3-1 and the other scenarios related to triangulation and trilateration are given in the next chapter below Section 6.2.

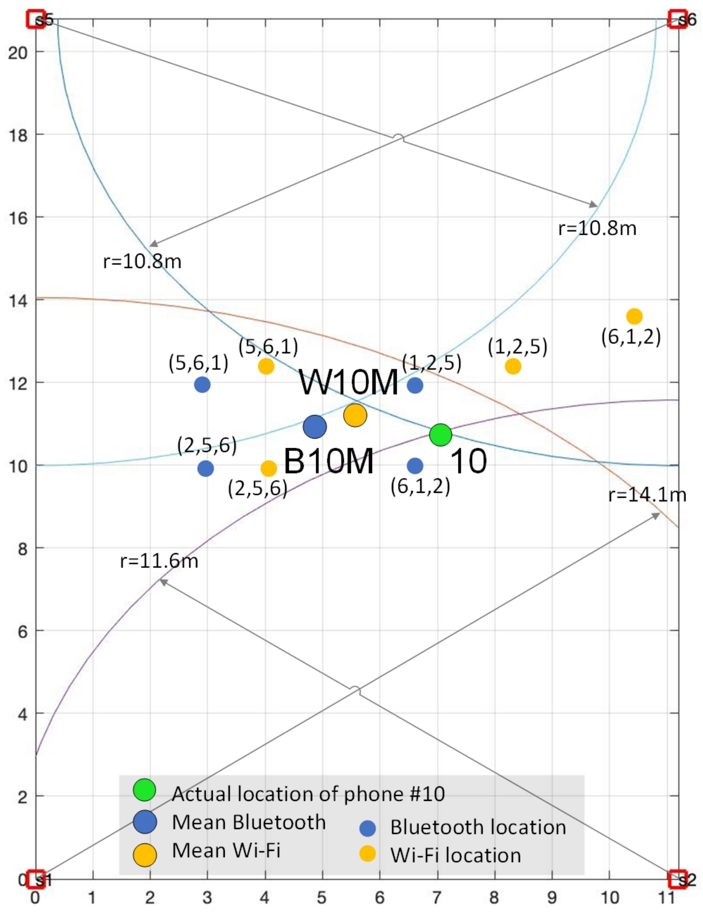


**Figure 5.6:** Trilateration - Estimated location of phone 10 based on different sensors and mean values from Wi-Fi and Bluetooth data

| | **Wi-Fi mean coordinates (x,y), distance to the actual location** | **Bluetooth mean coordinates (x,y), distance to the actual location** |
|---|---|---|
| Triangulation | (4.71, 11.02), 1.20 m | (5.70, 10.83), 2.20 m |
| Trilateration | (5.66, 11.16), 1.30 m | (4.89, 10.98), 2.02 m |
| Ground truth | (6.90, 10.78) | |

**Table 5.7:** Comparing the locations of phone 10 with the use of triangulation and trilateration based on Wi-Fi and Bluetooth data

Table 5.7 shows the accuracy of the estimations analysed above. The estimated position using trilateration based on the Wi-Fi data is, in this case, slightly more accurate than using the Bluetooth data or the triangulation method.

**Triangulation and trilateration uncertainty**

When it comes to various factors that affect estimating the devices' position, some need further attention. The first factor is multipath, where the signals propagate in different directions before it arrives the destination. This extra time because of multipath can allow other data packets transmitted later to be captured earlier at the sensor. Without techniques for getting the packets sorted in the correct sequence, multipath is the primary factor in inaccurate estimations. The second factor is related to the dynamic output power, as described in Section 5.5 about signal correlation. Here, the same challenge related to output power will occur with geolocation. The third factor is associated with SQL commands and the Matlab scripts. Although the programming part included in this master's thesis has been thoroughly tested, errors cannot be ruled out.

Chapter 6

# Results from signal correlation and geolocation methods

This chapter summarises the results of the signal correlation and the results of the geolocation methods using signal strength. The quality and reliability of the results are slightly touched upon but are covered later in the discussion section below 7.1.

When it comes to the signal correlation, the focus has been on showing the results from two static experiments (3-1 and 6-1), while the remaining results performed in experiment 3-2, 3-3, 6-2 and 6-3 are put in Appendix C. Due to a large amount of data, the results from triangulation and trilateration will only focus on some devices in the dynamic experiments. For the static experiments, all the devices are present in the same figures. Below is an overview of the results that are included in this chapter.

- Signal correlation
    - Linking devices in experiment 3-1 and 6-1 (all phones)
- Triangulation and trilateration using signal strength
    - Scenario 1 - Static, experiment 3-1 and 6-1 (all phones)
    - Scenario 4 - Dynamic, experiment 4 (phone 4, 5 and 10)
    - Scenario 4 - Dynamic, experiment 7-1 vs 7-2 (phone 5 and 7)
    - Scenario 5 - Dynamic, experiment 5 (phone 4, 5 and 10)

## 6.1 Signal correlation

In order to present the linking results, a top k-nearest neighbours approach have been used. The values from this approach are plotted in bar charts to show the different linking algorithms and the accuracy of the MAC addresses linked when top k was set to top 1, top 3 and top 5. When two signals are linked at top 1, the Bluetooth LAP and the Wi-Fi MAC address are linked in the first sorted row among the calculated Euclidean distances. The accuracy of this is related to the degree of correctness associated with the linking algorithm used. Even if the result has linked two MAC addresses, false positives (correct estimated, incorrect to ground truth) and false negatives resulting from poor programming can still occur.

### 6.1.1 Linking devices - Scenario 1 - Static, experiment 3-1

In experiment 3-1, ten devices were spread out to fixed locations for about three minutes. They were placed in an upright position on top of objects in their location. Human activity was restricted to a minimum and only while connecting to the Bluetooth device and starting the YouTube stream. In this static experiment, the goal was to see if signals would be linked and use the data for geolocation purposes. Figure 6.1 show the accuracy for different top k values which are coloured green (top 1), yellow (top 3) and orange (top 5). The y-axis shows different linking algorithms with information whether the method used four or six sensors. The algorithms that performed best in this experiment among the top 5 were (i) trilateration and (ii) Bluetooth to Wi-Fi conversion (6 sensors). In the top 5, both algorithms could link the MAC addresses with an 80 % accuracy, which is eight out of ten devices. Three algorithms could, with 40 % accuracy, link the correct MAC addresses on top 1. The different algorithms managed to link the same MAC addresses, but there were also differences. In any case, by comparing the algorithms, one can see indications of whether the signal correlation was performed correctly. Even in the top 5, one device was not linked at all. This device was positioned in the fixed location number seven, close to the long wall. Those devices that were frequently linked in the top 1 are listed as green in Table 5.6. The results also show that the algorithms using data from six sensors instead of four managed to link the devices slightly more accurately.

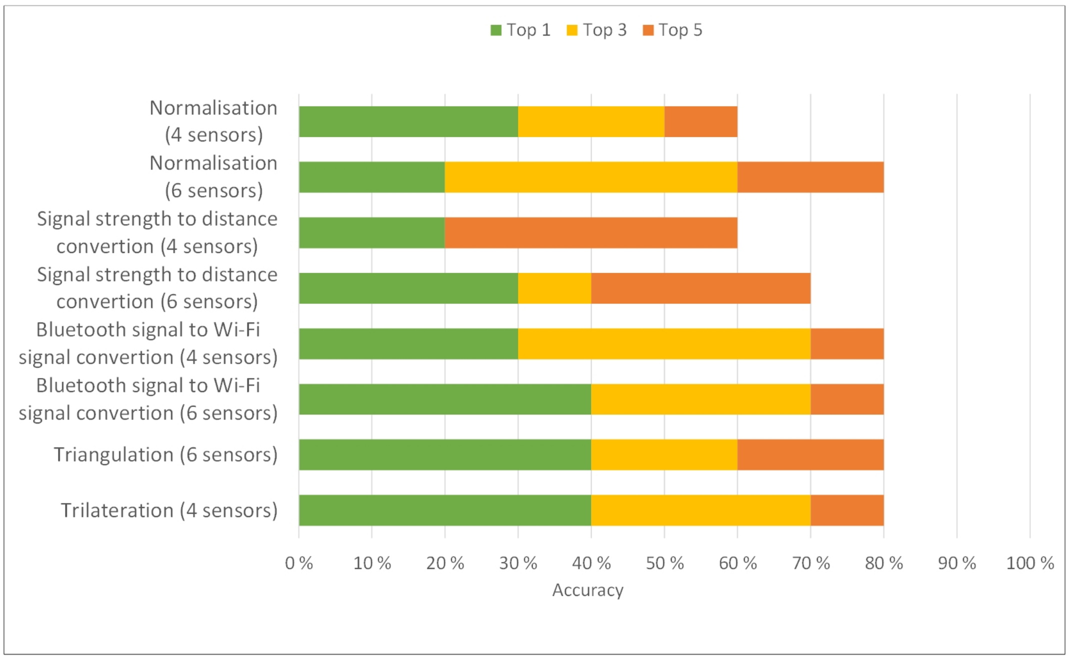


**Figure 6.1:** Top K nearest neighbours from experiment 3-1

### 6.1.2 Linking devices - Scenario 1 - Static, experiment 6-1

In experiment 6-1, data from ten devices were also collected. The devices were, this time, held by the participants in a comfortable and normal position in front of them. The participants were told to stand in an upright position and the exact location during the experiment. The goal of experiment 6-1 was to use the data for signal correlation and static geolocation. Even though the trilateration algorithm performed best with 80 % accuracy in the top 5, the algorithm using normalisation with four sensors linked most MAC addresses in top 1 (40 %). The correctness of the linking in top 1 made the normalisation algorithm appear more reliable than trilateration. Also, in this experiment, the device in location seven was never linked. Compared to Experiment 3-1, slightly fewer MAC addresses were linked in experiment 6-1. There is a noticeable difference when it comes to the number of MAC addresses that were linked in the top 1. The presence of several people in the room could have increased the probability of multipath. The absence of individual calibration of the devices can also be a factor of reduced accuracy.

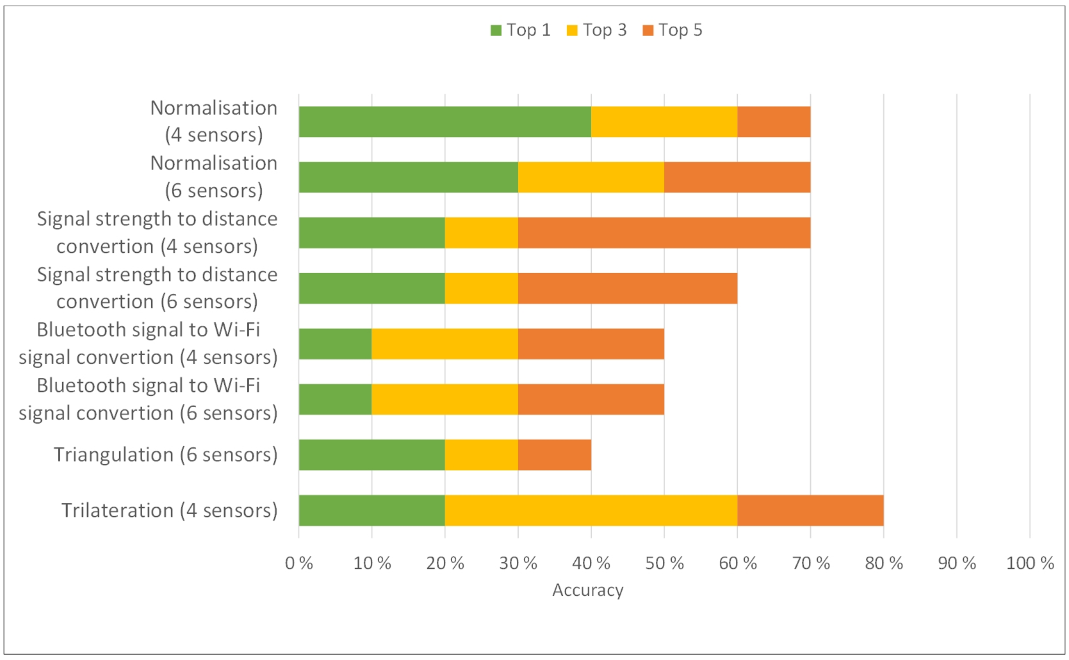


**Figure 6.2:** Top K nearest neighbours from experiment 6-1

## 6.2 Triangulation and trilateration using signal strength

The triangulation and trilateration result from different experiments is plotted in their respective Cartesian coordinate systems to present the accuracy. These coordinate systems illustrate the room where the experiments took place, where the x-axis and y-axis equal the width and height of the room in meters. The triangles in the next figures refer to the estimated mean value used by triangulation, while the circles refer to the estimated mean value of trilateration. These mean values originate from at least two estimations performed by different sensor pairs (triangulation) or sensor composition (trilateration) within the same second. The calculation behind these mean estimations is explained in the previous Chapter 5 and illustrated in Figure 5.5 (triangulation) and Figure 5.6 (trilateration). The estimated location with the dynamic experiments in mind also includes a timestamp represented in minutes and seconds. This timestamp relates to those signals that were compressed with the time-based filtering method. In order to distinguish between Wi-Fi and Bluetooth data, each estimate is identified with either a "W" er "B".

### 6.2.1 Geolocation - Scenario 1 - Static, experiment 3-1

Results from experiment 3-1 are presented in Figure 6.3. The goal was to pinpoint each device located in fixed positions to indicate how accurate the geolocation methods were. As we see, several estimates deviate from the actual locations.

Not many estimates are grouped either. However, some results stand out. One of these is the clustered estimates associated with phone 10. Both triangulation and trilateration have estimated that the device is located slightly to the left of the actual position. Another good result is the clustered group of phone 5 and phone 4. In addition, there are some perfect estimates, such as phone 2 estimated from the triangulation method based on Bluetooth data and the estimation of phone 5 based on trilateration.

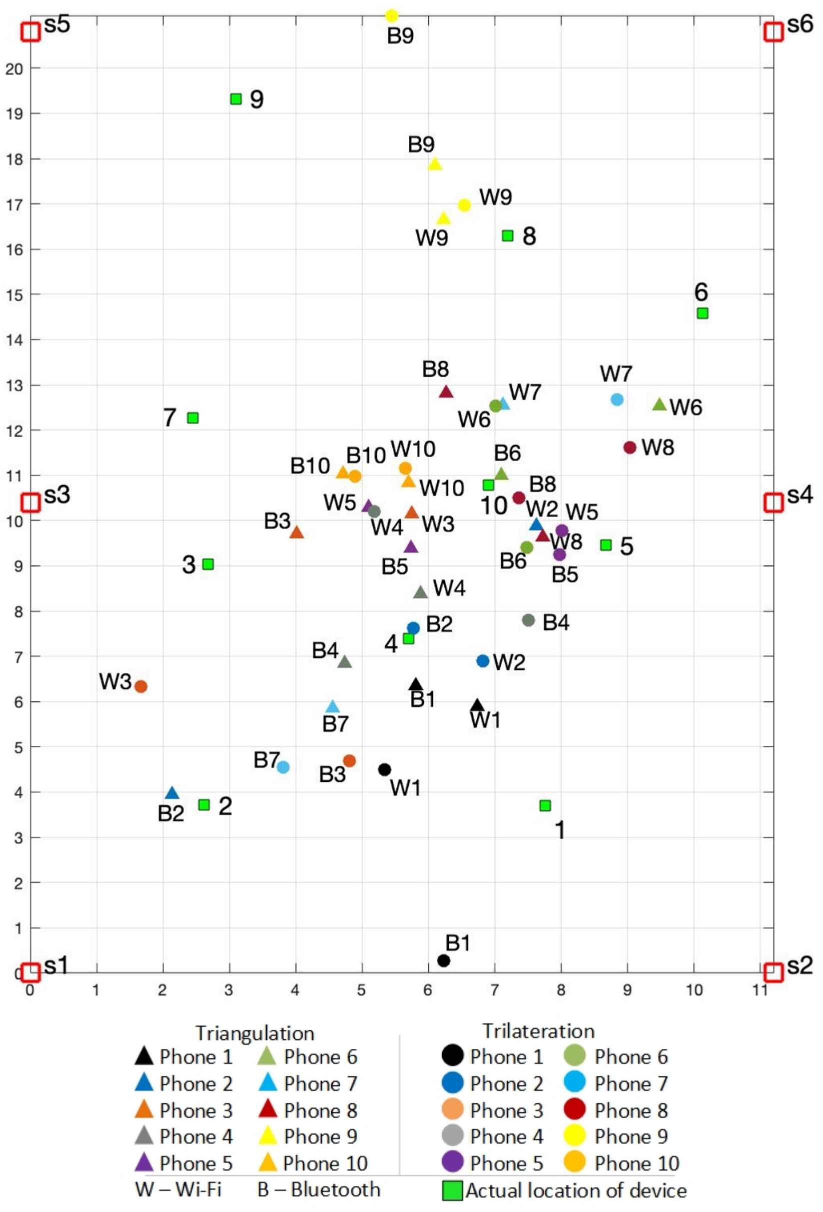


**Figure 6.3:** Experiment 3-1, static scenario with data from all known devices

### 6.2.2 Geolocation - Scenario 1 - Static, experiment 6-1

In experiment 6-1, the same goal as experiment 3-1 applies. Here, the devices were not individually calibrated but used a default output power instead. The results are scattered in Figure 6.4. We see that most of the estimates are more than 1.5 meters away from the actual position. Although some estimates are close to the actual position, they often have low reliability because they are not clustered with the other estimates from the same phone.

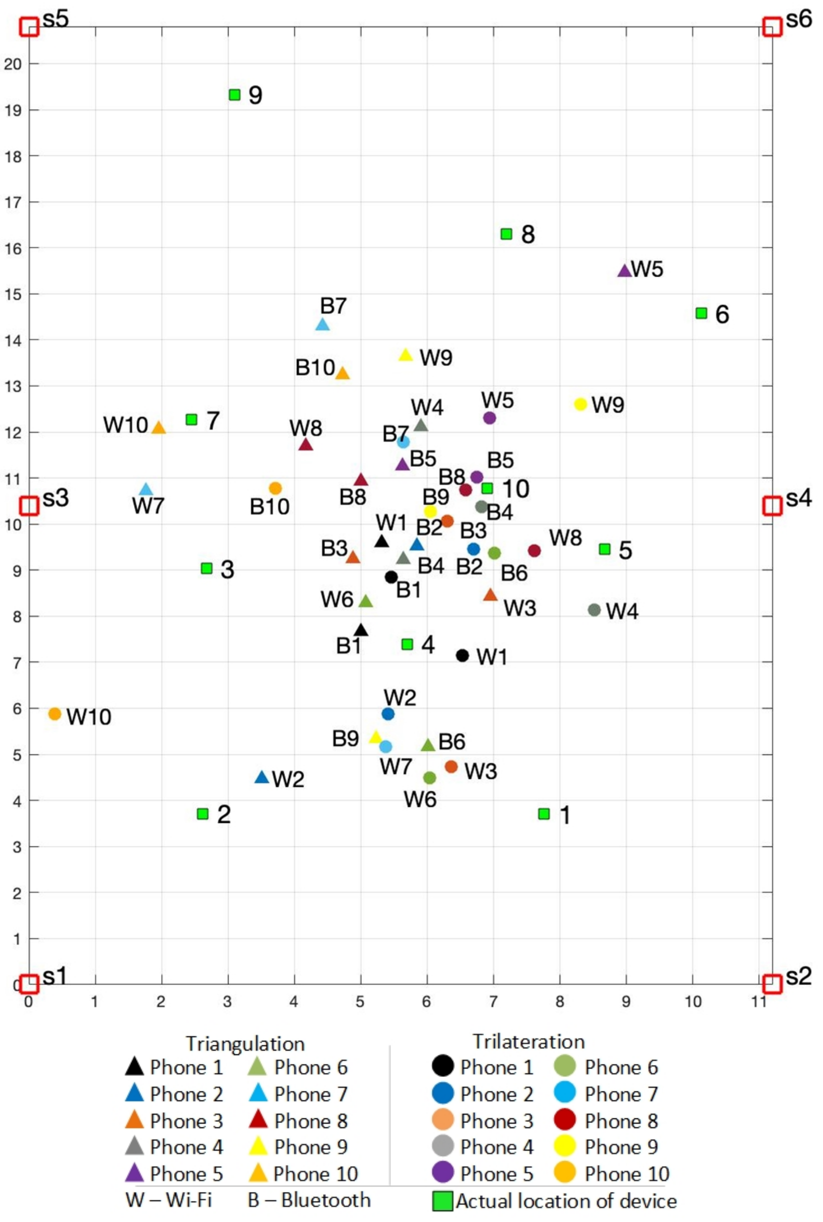


**Figure 6.4:** Experiment 6-1, static scenario with data from all unknown devices

### 6.2.3 Geolocation - Scenario 4 - Dynamic, experiment 4

In the first dynamic scenario, the devices should follow a pre-defined track. When arriving at position 2, 3, 5, 6 and 8, the devices should rest for 20 seconds. A theory was that the devices would be clustered around these positions. In order to present the results, three devices (phone 4, 5, 10) have been selected due to the high amount of data. The figures in the following pages refer to this experiment.

First, the estimated route of phone 4 is scattered in Figure 6.5. The geolocation result shows several estimates with the corresponding timestamp of when the data was captured. Immediately, there is no obvious pattern or clusters that can tell where the device was moving. When looking at the estimates from triangulation based on Wi-Fi data, it appears that these estimates are most accurate to the actual locations. In general, we can say that the device spent the most time in the centre, moving from the lower to the upper part of the room.

When it comes to phone 5, the estimated route is presented in Figure 6.6. Here, the estimates are less accurate. Some estimates of trilateration from Wi-Fi data appear as the most reliable. Nevertheless, the data confirm that the device has been in the room for some time. This piece of information can be important from a police perspective.

The estimated route of phone 10 is presented in 6.7. In general, the estimates of trilateration and triangulation show low accuracy. Although some estimates are reasonably correct, e.g. triangulation from Wi-Fi data, it is difficult to distinguish them from the remaining estimates from the same device with lower accuracy.

## Geolocation - Phone 4 in experiment 4

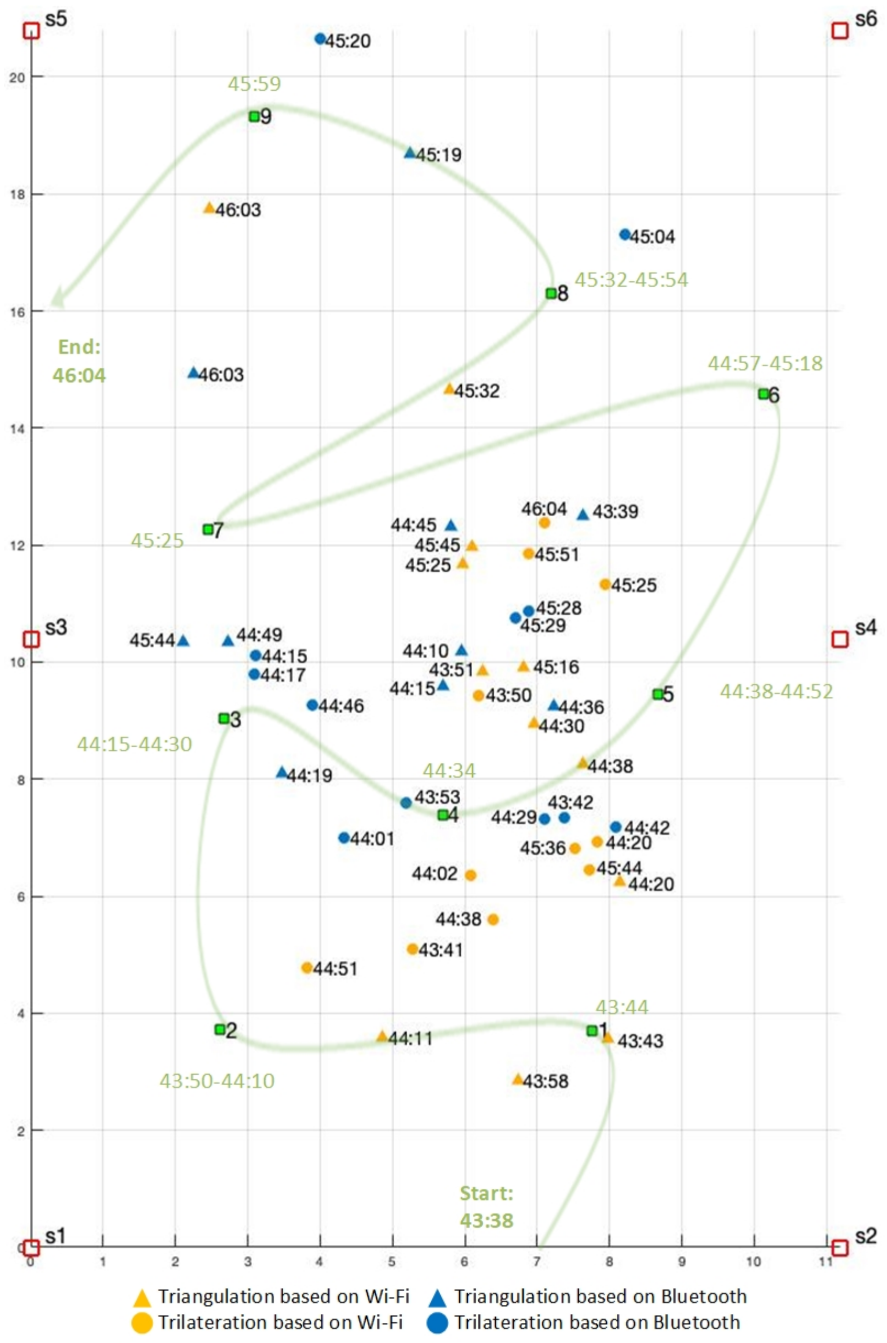


**Figure 6.5:** Results of phone 4 in experiment 4

### Geolocation - Phone 5 in experiment 4

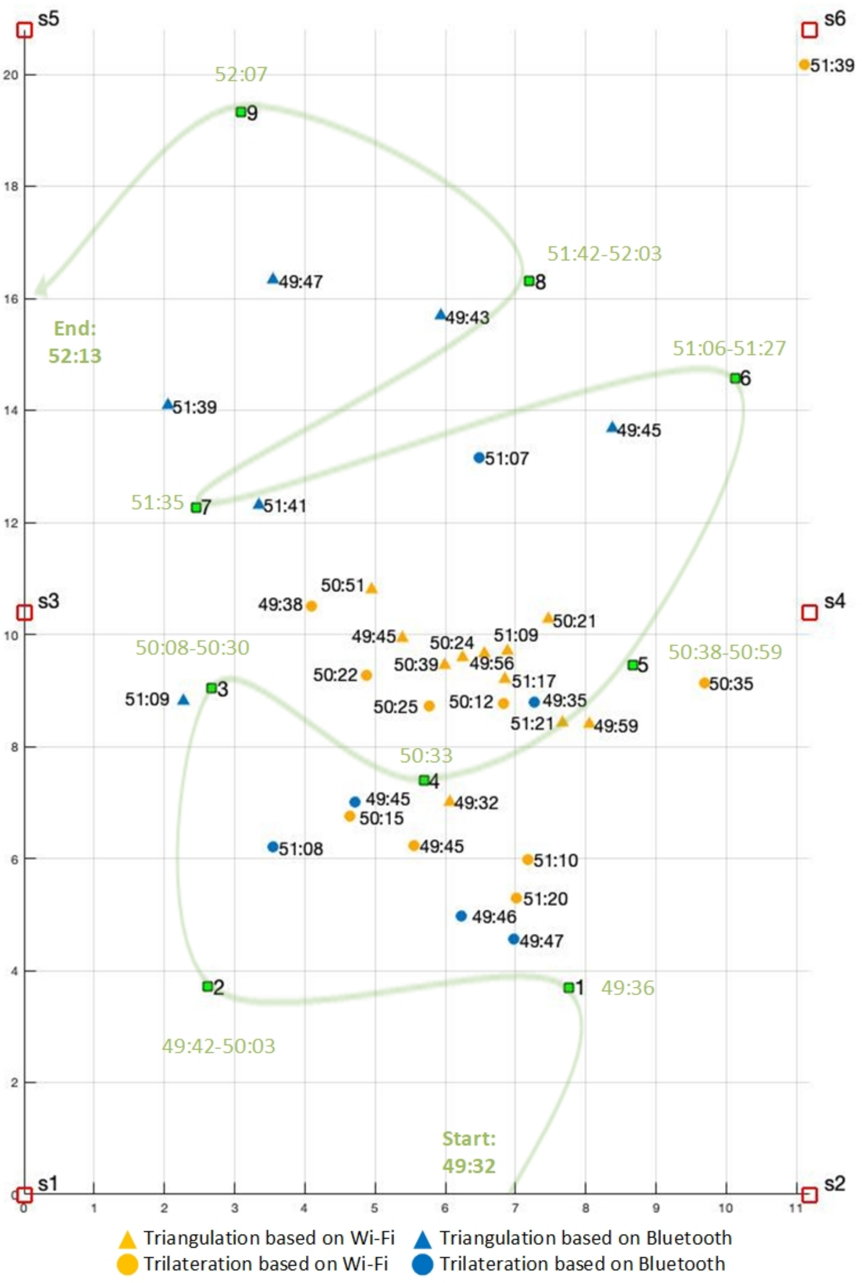


**Figure 6.6:** Results of phone 5 in experiment 4

## Geolocation - Phone 10 in experiment 4

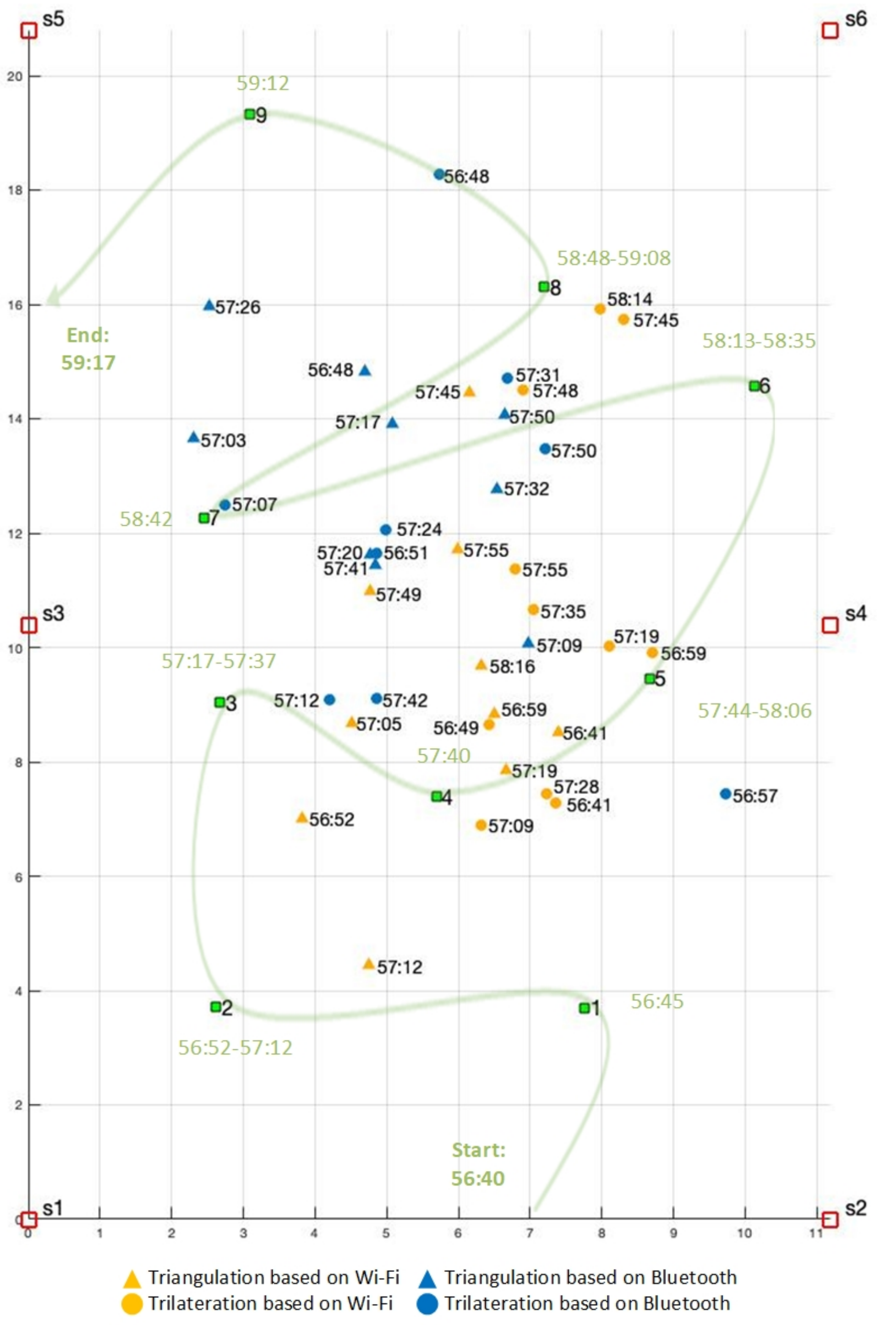


**Figure 6.7:** Results of phone 10 in experiment 4

### 6.2.4 Geolocation - Scenario 4 - Dynamic, exp. 7-1, 7-2 (phone 5,7)

Experiment 7-1 was performed in the same way as experiment 4, but this time, the group of devices was changed into the unknown devices that refer to the attending participants. To see if the participants' speed had any influence on the data, the participants walked about twice as fast in experiment 7-2 and 7-1. They rested for only three seconds in some locations (2, 3, 5, 6 and 8). The theory was that experiment 7-2 would give fewer estimates without any larger clusters around the locations as mentioned above.

First, Figure 6.8 show the results from experiment 7-1 with two selected phones (5 and 7). The figure shows a mixture of estimates with low and high accuracy between 1-6 meters, making the estimates together less reliable.

The estimated locations in experiment 7-2 of when the participants walked faster are shown in Figure 6.9. There were less than half as many mean estimates in this experiment compared to experiment 7-1 when the participants walked slower. That may explain some of the spread in this result compared to experiment 7-1. Although the participants walked faster, several estimates have high accuracy compared to actual location and time.

## Geolocation - Phone 5 and 7 in experiment 7-1

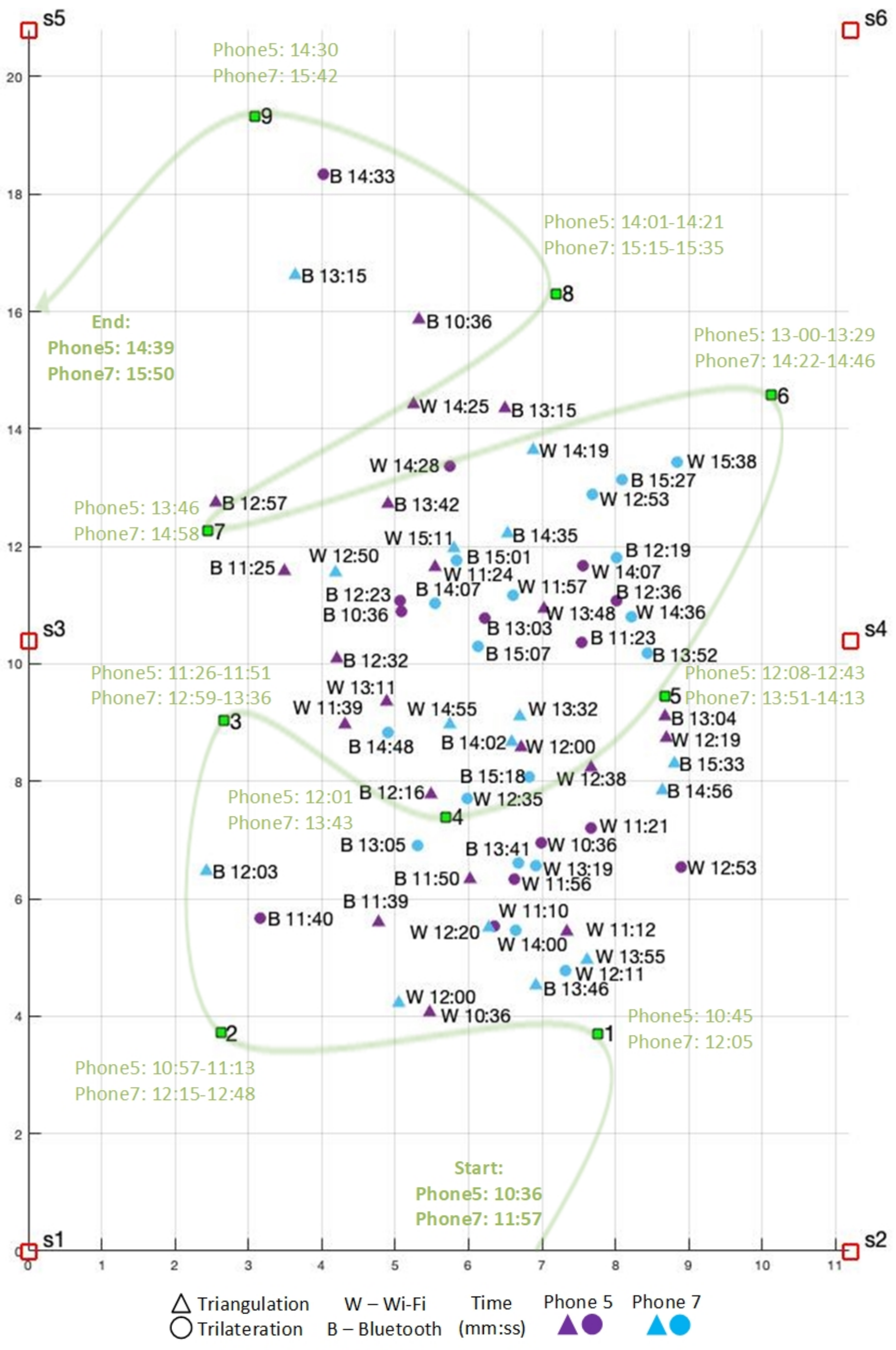


**Figure 6.8:** Results of phone 5 and 7 in experiment 7-1

## Geolocation - Phone 5 and 7 in experiment 7-2

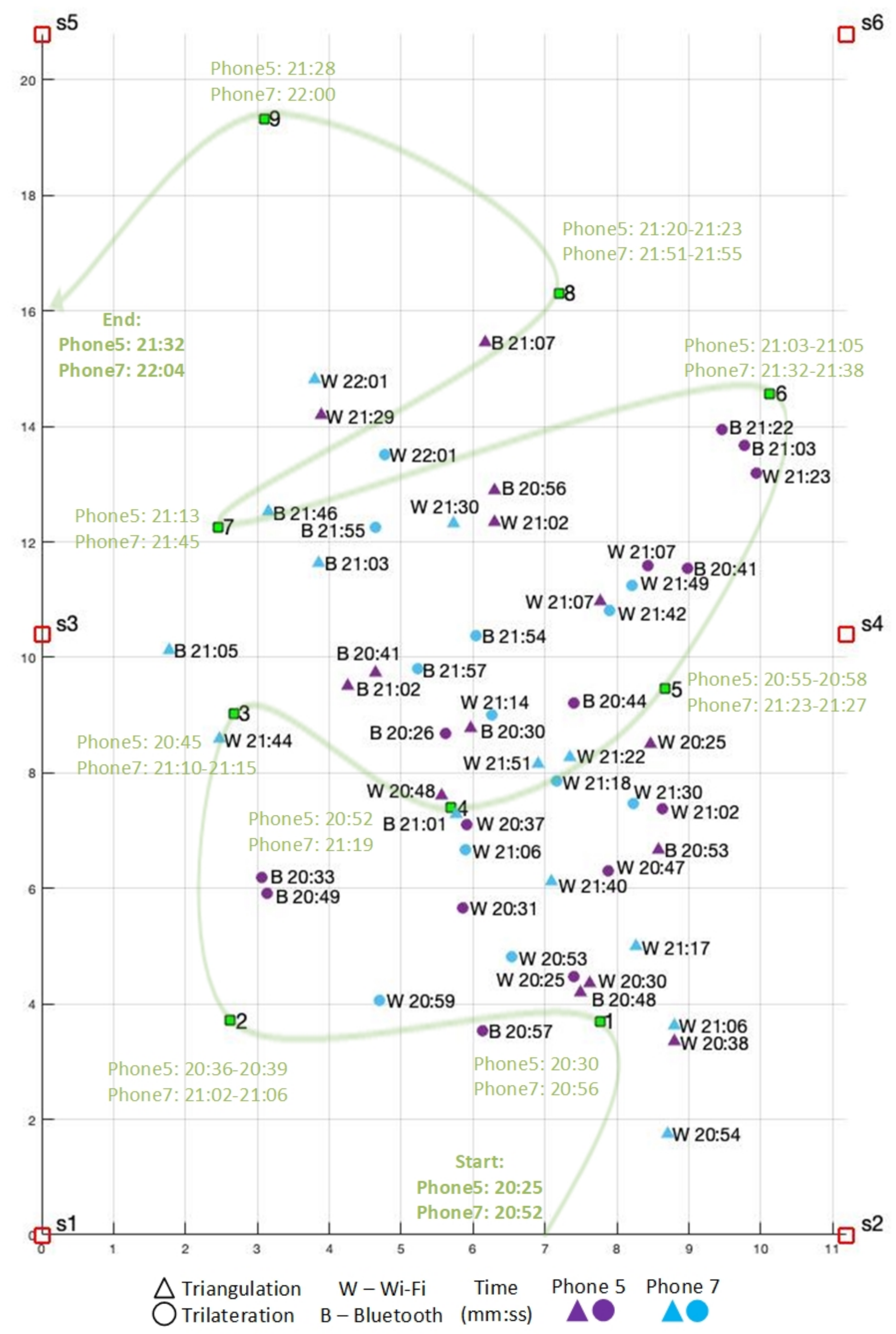


**Figure 6.9:** Results of phone 5 and 7 in experiment 7-2

### 6.2.5 Geolocation - Scenario 5 - Dynamic, exp. 5 (phone 4,5,10)

The last scenario was carried out in order to locate the devices when moving together as a group. Figure 6.10 shows three devices out of nine. Although the devices were close to each other along the way, it does not appear in the figure. Neither are the devices clustered around the locations where they rested for 20 seconds. Further, the estimated locations are not accurate if compared to the actual time and location. However, the triangulation results from Bluetooth and Wi-Fi data state that the devices were close to each other around "09:15". Also, in the time between "08:32-08:41", the results of trilateration/ triangulation from Bluetooth and Wi-Fi data state that the devices were near location 5. These results agree well with the ground truth.

## Geolocation - Phone 4, 5 and 10 in experiment 5

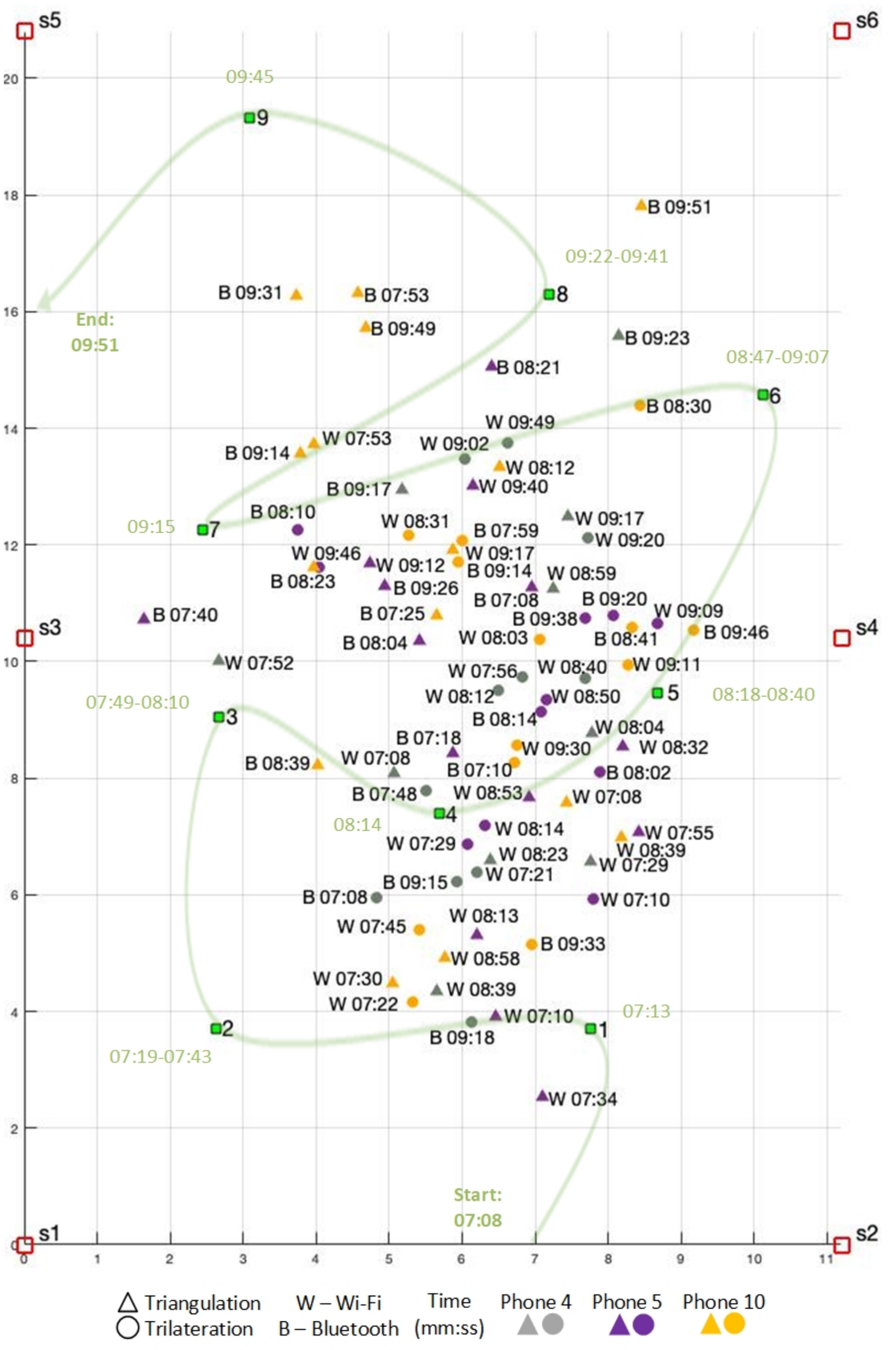


**Figure 6.10:** Results of phone 4, 5 and 10 in experiment 5

# Chapter 7

# Discussion and conclusion

This chapter includes a discussion of the results, the conclusion of this master's thesis and suggestions for further work. While the conclusion answers the main research question related to the problem statement, the discussion part discusses the sub-questions stated in Section 1.4 which were formulated by the main question. The discussion also includes our findings, strengths and limitations and possible use cases for law enforcement.

## 7.1 Discussion

### 7.1.1 General discussion about results and findings

The linking and geolocation methods are in this thesis based on signal strength, which means that the results are affected by the signal strength measured at each sensor. As for the Wi-Fi data, the signal strength varied a lot in the received TCP packets. These variations were the most surprising, as the network had a fixed channel, and the devices had almost a line of sight to each sensor. Here, it turned out that the Wi-Fi standard can dynamically adjust the transmitted power depending on the distance to the access point. We believe that this was one of the major factors that influenced the estimation of the devices' location. Although the devices were at rest in some experiments, the signal strength also varied in the Bluetooth data. These variations were also surprising since the distance between the Bluetooth devices and the mobile phone had almost the same distance throughout all of the experiments, which would indicate that the transmitted power would be constant. Thus, other factors affected the received Bluetooth signals. One of these could be the characteristic frequency hopping of Bluetooth.

We have tried to mitigate Bluetooth frequency hopping using all the channel's mean value in the calculation. Our tests show that various Bluetooth frequencies have a minor impact on the estimates. On that basis, the mean value of the frequencies was an acceptable solution.

Something that also could explain the varying signal strength in the received Bluetooth and Wi-Fi data is the interference from other signals. However, the effect of interference was not measured in this study due to a lack of special equipment. Nevertheless, according to research performed by Pei et al. [32] about mitigating interference (mentioned in Section 2.2), the Wi-Fi channel was set to a fixed frequency.

Another explanation of the signal strength inaccuracies may be the phenomenon of multipath caused by objects or people located nearby. In general, devices were more often estimated to be near the centre of the room. A theory could be that the locations closer to the walls were more challenging to estimate due to multipath. Also, some locations gave more deviations and were repetitive in several experiments. Among the results from the signal linking experiments (3-1 and 6-1), two different devices placed in location seven were never linked among the top five devices. The same trend for the same devices was seen in the results of geolocation. The inaccuracy shows that small changes, e.g. angle, rotation and minor location changes, would make a big difference indoors, even with multiple sensors covering different corners.

Suppose the multipath phenomenon is the main reason for reducing the accuracy of the estimated locations. In that case, the results should have been slightly better in the first set of the static geolocation experiments (3-1, 3-2 and 3-3) as there were only two persons in the room. In contrast, there were a total of 11 persons in the room during the second set of static geolocation experiments (6-1, 6-2 and 6-3). However, experiment 3-1 shows that the devices were spread more around the room but with device number seven estimated far away from its actual location. Experiment 6-1 (uncalibrated output power) shows the estimated locations generally gathered around the centre of the room. This observation may be due to the higher impact of multipath, uncalibrated output power or a combination of both.

Given the number of sensors used, the results of signal correlation show that it was better to use six sensors rather than four in most experiments. The increase to six sensors was especially valid in experiment 3-1, 3-2 and 3-3 (signal linking experiments with known devices) when each device was individually calibrated for output power according to actual distances. In experiment 6-1, 6-2 and 6-3 (signal linking experiments with unknown devices), in which the output power was based on a regression line from a more trustworthy device, it was challenging to say whether the increase from four to six sensors resulted in better accuracy. In general, the linking in these experiments without calibration of output power was less accurate.

One question that remains to be answered is whether the geolocation results would be more accurate if triangulation and trilateration were combined with Bluetooth and Wi-Fi as data sources. To answer this question, we use the static geolocation experiment 3-1 as an example. Looking at the result of experiment 3-1, it may seem that all ten locations would have been estimated with higher accuracy if the mean value of triangulation and trilateration from both Bluetooth and Wi-Fi were used. By merging these estimates, the results would also have been presented with less noise. For the dynamic experiments, however, estimating the locations using Bluetooth and Wi-Fi data would have been more challenging as data packets were received at different times.

Several interesting findings need to be mentioned. The first finding relates to mobile forensics and experiment number two (Apple Bleee). The Apple Bleee software provided screen status from Apple products, showing whether a person was actively using their phone or not. This piece of information is crucial to obtain an After first unlock (AFU) mobile extraction with encrypted information. In contrast, a Before first unlock (BFU) extraction (device restarted since last use) will only contain basic metadata. However, since the Bluetooth MAC addresses in the Apple Bleee software were randomised, it was not easy to distinguish between the devices from a fixed location.

Another interesting finding was that all the devices attending the experiments kept their random Wi-Fi MAC address over time. This connection between the device and the network would be helpful from a forensic perspective as the device can be linked to a specific place at a specific time.

### 7.1.2 Discussion about the research sub-questions

Four out of seven research sub-questions have already received attention in the previous chapters. These are linked to their corresponding sections below. The remaining research sub-questions (3, 4 and 6) are further discussed:

*1. What useful information can be collected passively from Bluetooth and Wi-Fi data?* Section 5.2 describes the data types and metadata used in this study. In general, the data types from Bluetooth and Wi-Fi were inspected in Wireshark. Only the most valuable and necessary data types were imported into SQL (i.e. signal strength, MAC address and timestamp). Compared to the Bluetooth packets, the Wi-Fi packets included more data.

*2. Given that both Bluetooth and Wi-Fi are enabled, how can one find that these signals originate from the same device?* This question was elaborated in Section 2.2. Previous research performed by Longo [6] states that Bluetooth and Wi-Fi could be correlated due to the same representation of the distance (between device and sensor). Based on this relation, we have tested five linking algorithms to see which that best could link the Bluetooth and Wi-Fi signals originating from the same device. We have seen that Wi-Fi is a more complex standard than first thought. Most likely, this is because we have captured actively TCP packets rather than the less frequent probe requests with constant output power.

*3. Which algorithms can be considered best to link Bluetooth and Wi-Fi signals originating from the same device?* This answer is based on the results presented in Section 6.1 and the results given in Appendix C. In general, the results are more accurate in the experiments where the devices were calibrated for output power. This calibration applies to the signal correlation experiments (3-1, 3-2 and 3-3), where trilateration gave the highest accuracy among the top 5 devices. However, the algorithm that gave the most reliable results was the conversion from Bluetooth to Wi-Fi signals (based on regression lines) using six sensors. This algorithm was more reliable as most devices were correctly linked in the top 1 (Bluetooth LAP and the Wi-Fi MAC address were linked in the first sorted row). In experiment 6-1, 6-2 and 6-3, where the devices were not calibrated for output power, the following algorithms gave the highest accuracy among the top 5 devices (i) normalisation of the received signal strength with six sensors and (ii) signal to distance with six sensors. Nevertheless, the most reliable algorithm in linking devices correctly in top 1 was trilateration.

*4. Which geolocation algorithms can be considered best to track devices using signal strength from Bluetooth and Wi-Fi data?* This answer is based on the results from the geolocation experiments presented in Section 6.2. The results show that triangulation is slightly more accurate than trilateration when looking at the static geolocation experiments. Also, the results are more clearly shown in the static experiment 3-1 than 6-1 because of the output power calibration. It is difficult to determine what technology that best can estimate the location - the accuracy seems to be quite similar, with variations around 1-7 meters.

In the dynamic experiments, the estimations from the use of triangulation also seem more accurate. However, it is difficult to determine what technology that best can estimate each device's movement pattern. For instance, Bluetooth appears more accurate in one part of the room, while Wi-Fi appears more accurate in other parts. In this case, a mean value of the triangulation results based on Wi-Fi and Bluetooth could provide better estimations. The same goes for the results performed with trilateration.

*5. Can signal interference be a problem while collecting data?* This was elaborated in Section 2.2. The influence of interference was not measured during this study.

*6. What technical challenges may arise when the police collect data from Bluetooth and Wi-Fi?* There are two important factors to mention when working with wireless signals at these short distances. First, the area where the data from Bluetooth and Wi-Fi are captured should include a video surveillance camera. Alternatively, a person with mobile equipment will perceive what is happening "on the go" instead of a camera. Secondly, a correct time source must be visible in the recorded video. These two factors were helpful in this study to confirm actual position and time. The process of finding ground truth would have become a technical challenge if they were not accounted for. As mentioned in Section 4.3.2, a challenge occurred during the experiment due to a large amount of Wi-Fi data. A future sensor network should, on this basis, include mechanisms that filter the data continuously during the collection. Also, sufficient network speed is necessary when the amount of data increases.

*7. How should irrelevant and misleading data be filtered out?* In general, we have filtered out unwanted observations using whitelisting (based on MAC addresses), channel and time. In addition, we have performed distance-based filtering to remove weak/strong signals outside the thresholds calculated in Section 5.3. We have also performed time compression on the data collected in the dynamic scenarios to obtain simpler timestamps (seconds rather than milliseconds). Here, the signal values within the same second were compressed into a mean value associated with the new timestamp. Section 3.4 gave a brief introduction to the filtering and cleaning methods, while Section 5.3 described in-depth how these filters were applied.

### 7.1.3 Strengths and limitations of the study

In this master's thesis, we have captured data from Bluetooth and Wi-Fi based on two different methods. The method of capturing Bluetooth data did not require any user input besides that the device had to stream music with the Bluetooth Classic standard. In contrast, when capturing Wi-Fi data, the participant needed to connect to a wireless network. The accessibility of Bluetooth data implies that this technology is better suited for outside tracking. However, it has become common for people in public places to access free Wi-Fi, e.g., shopping malls, airports, and public transport. Perhaps, in a smart city setting, we may see Wi-Fi become more available and free?

Because devices are manufactured with different chipsets, they transmit signals differently. This factor is a disadvantage in this study because it is advantageous to calibrate the received signal values to get a more accurate result. A solution to this real-life challenge would be to compare signal strength to hidden on-the-fly devices where the distance is known. Although the results show that it is better to calibrate individual devices based on their signal strength when the distance is known, the result can still be satisfying for the police.

We are proud to say that the sensor network in this study worked nearly perfectly and as intended. This achievement enabled us to manage sensors securely through a VPN tunnel over the internet by distributing Ansible commands to multiple sensors in parallel. The architecture in combination with the Ansible framework is to be recommended in similar projects.

The script part in Matlab is, in this study, a prototype, and cognitive biases can therefore not be ruled out. If the study had been repeated, several persons should have reviewed the programming more carefully to exclude any mistakes.

A limitation is related to how the data was processed in the dynamic experiments. Due to the phenomenon of multipath and interference, an unique data packet can be captured at different times by the sensors. This interference will result in unordered sequences of data packets, making it difficult to identify data packets without any identifiers. Since the Bluetooth data did not contain an identifier, the order of these data packets was based on the time when the signals were received. When it comes to Wi-Fi, the data packets contained an incremental packet number. However, it was rare for all six sensors to capture the same data packet. This observation was the reason why the packet number lost our interest in this study.

### 7.1.4 Possible use cases for law enforcement

So, how can law enforcement make use of the results and findings from this study? The police are already able to analyse Bluetooth logs from mobile extraction to find, among other things, the connection between a Bluetooth device and the paired mobile phone. Wi-Fi is also often used by the police to link a device to a crime scene. Nevertheless, the possibilities are many, and some of these are discussed below.

#### Map devices with Bluetooth and Wi-Fi in important areas or special situations

The most obvious way to make use of a sensor network is to map devices in a specific area as a live feed. A permanent sensor would also give historical activity with devices' frequency and function as an alarm that can trigger specific devices. This concept would also be able to estimate the direction of a device and confirm or disprove any device presence. The latter could become a database for investigators to enrich their police reports. Another possibility is to discover the cooperation of people based on geolocation and link the devices to a specific location in time. From a telecommunication perspective, this is not new. It is already possible to obtain "traffic data" from base stations containing valuable logs from SMS, calls and network use. However, logs from Bluetooth and Wi-Fi may still be interesting if the device is in flight mode or serve as a supplement to traffic data. Furthermore, a discovered MAC address from a sensor network would be easy to link to a database that holds information from existing mobile extractions that may provide the full identity of the device's owner.

#### Link Bluetooth and Wi-Fi data to combine metadata from the same device

The investigators could interpret more metadata by linking a random Wi-Fi MAC address with a unique Bluetooth LAP address originating from the same device. The idea is to use Bluetooth or Wi-Fi to answer investigative questions limited by the other data source. Together, the artefacts may increase situational awareness for investigators and give higher reliability when performing geolocation. If the geolocation methods based on Bluetooth and Wi-Fi are interpreted equally, the credibility in the court of law would also increase.

#### Capture Bluetooth and Wi-Fi data with mobile sensors

A fact is that the police sometimes struggle to identify people that have not behaved properly. It would be beneficial to bring mobile sensors to collect Bluetooth and Wi-Fi data (only probe requests as the devices are not connected to a wireless network) from public demonstrations (e.g. SIAN). An overview of, e.g. MAC addresses and device names, could give investigators a greater level of situational awareness in situations where people may be attempting to remain anonymous.

**Stalker attack - obtain correct data from a specific device on the move**

It can be challenging to distinguish devices from each other in larger gatherings and link a specific device to the correct individual. The idea is to let the police use mobile equipment to capture Bluetooth data while pursuing an individual. This method would increase the possibility of obtaining the correct device name and MAC/LAP address. The same equipment (i.e. Raspberry Pi and antennas) and software (Blue Hydra, Ubertooth or Apple Bleee) utilised in this study may be used for this purpose. When the correct MAC/LAP address is found, it would be efficient to search among existing mobile extractions to reveal any identities. The method must consider that a person may have several devices and that these devices may be in several places simultaneously (e.g. smartphone, smartwatch and laptop).

## 7.2 Conclusion

This master's thesis was set off with this research question: *How can the police use Bluetooth and Wi-Fi data for tracking and identification in covert operations?*

In search of an answer to this question, we have focused on signal correlation and geolocation. Signal correlation involves the linking of Bluetooth and Wi-Fi signals that belongs to the same device. Here, the intention was to combine metadata and find the connection between a device's Wi-Fi MAC address (or its random Wi-Fi MAC address) and its Bluetooth Lower Address Part. Geolocation means, in this case, to estimate the location of a device using triangulation or trilateration based on the received signal strength. To achieve this, we created a sensor network that was able to capture Bluetooth and Wi-Fi data. The collected data was analysed and imported into a SQL database. A connection from the database enabled us to perform calculations and visualisations directly in Matlab.

Our results show that it is possible to perform a satisfying signal correlation when calibrating the received signal values based on device-specific output power. Following the Euclidean distance (metric function) and the top k-nearest approach, we were able to find the signal pairs (ergo MAC addresses) closest linked to each other. The MAC address pair with the lowest value gave the best linking. Among the five algorithms tested, the most reliable algorithm was the conversion from Bluetooth to Wi-Fi signal. This algorithm was able to link the MAC addresses correctly with 29-40 % accuracy. Among the three best signal pairs linked, it was between 43-70 % probability that the signal pair derived from the same device. Among the five best signal pairs, the accuracy increased between

57-80 %.

Without calibration, the most reliable algorithm was trilateration that correctly linked the MAC addresses with 20-30 % accuracy. Among the three best signal pairs linked, it was between 40-60 % probability that the signal pair derived from the same device. Among the five best signal pairs, the accuracy increased between 50-80 %. In general, the accuracy improved for the algorithms that used six sensors rather than four.

The results of geolocation show that the triangulation method performs slightly better than the trilateration method. However, both methods estimate within the same accuracy; 1-7 meters from the actual location. The devices calibrated for output power had also their estimations better grouped around the actual location. Although several devices were estimated well concerning their position, some were less accurate - these non-accurate results derived from the dynamic geolocation experiments. The degree of accuracy from the dynamic experiments was not expected and made it hard to show that the devices followed a pre-defined route. Also, the varying results from different devices made it difficult to see whether it was Bluetooth or Wi-Fi data that estimated the position best.

With metadata from Bluetooth and Wi-Fi combined and the ability to track devices within seven meters accuracy, law enforcement may increase their situational awareness in cyberspace (and real life). Specifically, it will be appropriate to search for historical activity among Bluetooth and Wi-Fi data to link a device (and its owner) to a specific area at a precise time. The direction in which the device was moving can also be useful information for police investigators.

## 7.3 Further work

Future work may include a deeper analysis of data packets, suggestions for new methods or improvement in those methods that have already been tested in this thesis. New ideas and directions relevant to the topic are described below.

#### Look into the new Bluetooth Low Energy standard v5.2

During this master's thesis, the next generation of Bluetooth has emerged. The new version called Bluetooth 5.2 with LE audio is based on the Low Energy protocol and is predicted to take over for Bluetooth Classic currently used for audio transferring. The new standard would be exciting to look into. Also, other wireless technologies which are associated with the internet of things may be deemed relevant, e.g. Zigbee, Z-wave or the new smart home standard; Matter.

#### Identify the same data packets on several sensors

In order to achieve more accurate results with the geolocation methods used in this thesis, it would be beneficial to take a closer look at the packet number in each Wi-Fi packet. Instead of taking the mean value of all the signal strength within the exact second, one can focus on the signal values from the same packet number. The correct order of packets would prevent multipath signals from degrading the average value.

### Increase the results in general

To further increase the accuracy of the signal correlation, it would be possible to compare the algorithms to confirm similarities or differences. When it comes to the dynamic geolocation results, the estimated locations may become more reliable if Bluetooth and Wi-Fi were combined. It may also be relevant to combine the estimated locations based on trilateration and triangulation.

### Improvements in the architecture of the sensor network

In the collection phase, we discovered a challenge related to collecting large amounts of Wi-Fi data. A temporary solution was to store the data locally at each sensor. Hence, it would be necessary to improve the network flow further. An automated filtering process will also be necessary if the sensor network is used on a larger scale. In addition, the Matlab script should be updated to be more straightforward and more automated so that it can be compiled into an executable program.

### Rotate the devices in different fixed location when they are at rest

For the static experiments, it would be interesting to compare the estimated locations when the devices are rotated between the fixed locations. The result can tell if the geolocation methods provide the exact location for different devices.

### Present the estimated locations in a better fashion

In collaboration with the project partner, an idea has been to analyse and present the estimated location from geolocation in a better fashion. A broader illustration of the potential system architecture is found in appendix B. The system architecture includes a web server running Kafka, which can receive and process collected data from the sensor nodes. Elasticsearch (back end) and Kibana (front end) can also be used as tools to put information into context and provide better structure for the police.

# Bibliography

[1] H. Kolberg, *Gaining situational awareness using wi-fi*, Master's thesis, Norwegian University of Science and Technology, 2020.

[2] C. Groba, 'Demonstrations and people-counting based on wifi probe requests', in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, pp. 596–600. DOI: `10.1109/WF-IoT.2019.8767208`.

[3] C. Chilipirea, A. Petre, C. Dobre and M. van Steen, 'Presumably simple: Monitoring crowds using wifi', in *2016 17th IEEE International Conference on Mobile Data Management (MDM)*, vol. 1, 2016, pp. 220–225. DOI: `10.1109/MDM.2016.42`.

[4] metageek, *Understanding rssi*, Retrieved from `https://www.metageek.com/training/resources/understanding-rssi.html` [Online; accessed 2021-02-23], 2021.

[5] J. Marcel, *3 key factors that determine the range of bluetooth*, Retrieved from `https://www.bluetooth.com/blog/3-key-factors-that-determinethe-range-of-bluetooth/` [Online; accessed 2021-02-23], 2019.

[6] E. Longo, *Pairing wi-fi and bluetooth mac addresses through passive packets capture*, Retrieved from `https://www.politesi.polimi.it/bitstream/10589/138894/3/el-tesi.pdf` [Online; accessed 2020-04-06], 2017.

[7] C. Ansley, *Mac randomization in mobile devices*, Retrieved from `https://www.nctatechnicalpapers.com/Paper/2019/2019-mac-randomization-in-mobile-devices` [Online; accessed 2020-04-13], 2019.

[8] J. K. Becker, D. Li and D. Starobinski, 'Tracking anonymized bluetooth devices', *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 50–65, 2019.

[9] M. Afaneh, *Bluetooth addresses & privacy in bluetooth low energy*, Retrieved from `https://www.novelbits.io/bluetooth-address-privacy-ble/` [Online; accessed 2021-02-28], 2020.

[10] CISCO, *What is wi-fi?*, Retrieved from `https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html` [Online; accessed 2021-02-23], 2021.

[11] G. Phillips, *The most common wi-fi standards and types, explained*, Retrieved from https://www.makeuseof.com/tag/understanding-common-wifi-standards-technology-explained/ [Online; accessed 2021-02-23], 2021.

[12] C. David, *2.4 ghz channel planning*, Retrieved from https://www.extremenetworks.com/extreme-networks-blog/2-4-ghz-channel-planning/ [Online; accessed 2021-02-23], 2021.

[13] Blumira, *Promiscuous mode*, Retrieved from https://www.blumira.com/glossary/promiscuous-mode/ [Online; accessed 2021-02-23], 2021.

[14] Github, *Radiotap*, Retrieved from https://www.radiotap.org/ [Online; accessed 2021-02-23], 2021.

[15] R. Triggs, *What is le audio and lc3, the latest in bluetooth audio?*, Retrieved from https://www.soundguys.com/bluetooth-le-audio-lc3-explained-28192/ [Online; accessed 2021-02-26], 2020.

[16] N. D, *Bluetooth 5.0 vs. 4.0 - a detailed comparative analysis*, Retrieved from https://www.intuz.com/blog/bluetooth-version-5-vs-4-comparative-analysis [Online; accessed 2021-02-26], 2020.

[17] E. Ferro and F. Potorti, 'Bluetooth and wi-fi wireless protocols: A survey and a comparison', *IEEE Wireless Communications*, vol. 12, no. 1, pp. 12–26, 2005. DOI: 10.1109/MWC.2005.1404569.

[18] J. Marcel, *How bluetooth technology uses adaptive frequency hopping to overcome packet interference*, Retrieved from https://www.bluetooth.com/blog/how-bluetooth-technology-uses-adaptive-frequency-hopping-to-overcome-packet-interference/ [Online; accessed 2021-02-26], 2020.

[19] Argenox, *Introduction to bluetooth classic*, Retrieved from https://www.argenox.com/library/bluetooth-classic/introduction-to-bluetooth-classic/ [Online; accessed 2021-02-26], 2020.

[20] M. Davies, E. Furey and K. Curran, 'Improving compliance with bluetooth device detection', *Telkomnika*, vol. 17, no. 5, pp. 2355–2369, 2019.

[21] GitHub, *Bluehydra*, Retrieved from https://github.com/pwnieexpress/blue_hydra [Online; accessed 2020-04-14], 2020.

[22] LCDI, *Bluetooth tracking analysis*, Retrieved from https://www.champlain.edu/Documents/LCDI/Bluetooth%20Tracking%20Analysis_2017.docx.pdf [Online; accessed 2020-04-15], 2020.

[23] L. Bai, N. Ireson, S. Mazumdar and F. Ciravegna, 'Lessons learned using wi-fi and bluetooth as means to monitor public service usage', in *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, 2017, pp. 432–440.

[24] L. Schauer, M. Werner and P. Marcus, 'Estimating crowd densities and pedestrian flows using wi-fi and bluetooth', in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2014, pp. 171–177.

[25] M.-H. Tsai, J.-N. Luo, M.-H. Yang and N.-W. Lo, 'Location tracking and forensic analysis of criminal suspects' footprints', in *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, IEEE, 2019, pp. 210–214.

[26] Hexway, *Apple bleee. everyone knows what happens on your iphone*, Retrieved from `https://hexway.io/research/apple-bleee/` [Online; accessed 2020-04-16], 2019.

[27] F. Ryan and M. Schukat, 'Wi-fi user profiling via access point honeynets', in *2019 30th Irish Signals and Systems Conference (ISSC)*, IEEE, 2019, pp. 1–4.

[28] A. O. S. Project, *Privacy: Mac randomization*, Retrieved from `https://source.android.com/devices/tech/connect/wifi-mac-randomization` [Online; accessed 2020-04-14], 2020.

[29] Apple, *Use private wi-fi addresses in ios 14, ipados 14 and watchos 7*, Retrieved from `https://support.apple.com/en-au/HT211227` [Online; accessed 2021-02-21], 2021.

[30] Apple, *Wi-fi privacy*, Retrieved from `https://support.apple.com/guide/security/wi-fi-privacy-secb9cb3140c/web` [Online; accessed 2021-02-24], 2021.

[31] J. K. Becker, D. Li and D. Starobinski, 'Tracking anonymized bluetooth devices', *Proceedings on Privacy Enhancing Technologies*, pp. 50–65, 2019.

[32] L. Pei, J. Liu, Y. Chen, R. Chen and L. Chen, 'Evaluation of fingerprinting-based wifi indoor localization coexisted with bluetooth', *The Journal of Global Positioning Systems*, vol. 15, no. 1, p. 3, 2017.

[33] L. Schauer, F. Dorfmeister and M. Maier, 'Potentials and limitations of wifi-positioning using time-of-flight', in *International Conference on Indoor Positioning and Indoor Navigation*, IEEE, 2013, pp. 1–9.

[34] B. O'Keefe, 'Finding location with time of arrival and time difference of arrival techniques', *ECE Senior Capstone Project*, 2017.

[35] M. H. Sarshar, 'Analyzing large scale wi-fi data using supervised and unsupervised learning techniques', PhD thesis, Dalhousie University, 2017.

[36] M. Woolley, *Bluetooth direction finding: A technical overview*, Retrieved from `https://www.bluetooth.com/bluetooth-resources/bluetooth-direction-finding/` [Online; accessed 2020-04-20], 2019.

[37] A. Kurkcu and K. Ozbay, 'Estimating pedestrian densities, wait times, and flows with wi-fi and bluetooth sensors', *Transportation Research Record*, vol. 2644, no. 1, pp. 72–82, 2017.

[38] Wireguard, *Protocol & cryptography*, Retrieved from `https://www.wireguard.com/protocol/` [Online; accessed 2021-02-19], 2021.

[39] elitedatascience, *Data cleaning*, Retrieved from `https://elitedatascience.com/data-cleaning` [Online; accessed 2021-02-07], 2021.

[40] H. T. Friis, 'A note on a simple transmission formula', *Proceedings of the IRE*, vol. 34, no. 5, pp. 254–256, 1946. DOI: `10.1109/JRPROC.1946.234568`.

[41] ElectronicsNotes, *Free space path loss: Details & calculator*, Retrieved from `https://www.electronics-notes.com/articles/antennas-propagation/propagation-overview/free-space-path-loss.php` [Online; accessed 2020-10-11], 2019.

[42] J. Du, J.-F. Diouris and Y. Wang, 'A rssi-based parameter tracking strategy for constrained position localization', *EURASIP Journal on Advances in Signal Processing*, vol. 2017, no. 1, pp. 1–10, 2017.

[43] H. Anton and C. Rorres, *Elementary linear algebra: applications version*, 11th ed. John Wiley & Sons, 2013, p. 145.

[44] S. Raschka, *About feature scaling and normalization*, Retrieved from `https://sebastianraschka.com/Articles/2014_about_feature_scaling.html` [Online; accessed 2020-04-21], 2014.

[45] S. Garcia-Villalonga and A. Perez-Navarro, 'Influence of human absorption of wi-fi signal in indoor positioning with wi-fi fingerprinting', in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2015, pp. 1–10. DOI: `10.1109/IPIN.2015.7346778`.

[46] accoladewireless, *Why wifi is complicated: Wifi signal issues*, Retrieved from `http://www.accoladewireless.com/wlan-wifi-signal-issues/` [Online; accessed 2021-02-11], 2016.

[47] Re: Wi-Fi signals and transmit power [Questioned in an online digital forensics forum, network-forensics, 2020-11-27].

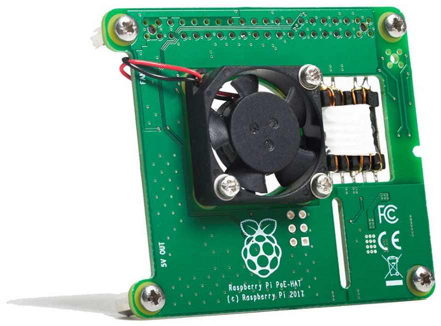
# Appendix A

# Equipment

## A.1 Main equipment

- Raspberry Pi 4 Model B 4GB Starter Kit
    - Raspberry Pi 4 Model B, 4GB RAM
    - Raspberry Pi 4 3A USB-C Charger, Black
    - SanDisk MicroSDHC Ultra 32GB
    - Raspberry Pi 4 Model B Case, Black
- PoE hat for Raspberry Pi 3B and 4B
- ASUS USB-BT500 Bluetooth USB-adapter
- Ubertooth One w/ case
- Alfa Network AWUS036ACH

**Figure A.1:** Raspberry Pi 4 Model B 4GB Starter Kit



**(a)** PoE hat for Raspberry Pi 3B and 4B

**(b)** Ubertooth One USB adapter

**(c)** ASUS BT500 Bluetooth USB adapter

**(d)** Alfa Network AWUS036ACH USB adapter

**Figure A.2:** Additional tools used with each Raspberry Pi

## A.2 Extra equipment

- GoPro camera
- 10x smartphones (table A.3)
- 10x Bluetooth devices
- Wi-Fi router (TP-Link, Archer C7)
- 2x clients (time source and controlling node)

## A.3 Smartphones used in the experiments

Devices in table A.1 were collected from friends, family and work. They are grouped as "known devices", meaning they were subjected to additional testing over time. Further on, devices in A.2 were the "unknown devices" that associates owned.

| # | **Brand** | **Model** | **Operating system** |
|---|---|---|---|
| 1 | Samsung S20 | SM-G981B/DS | Android OS 10 |
| 2 | Samsung S8 | SM-G950F | Android OS 9 |
| 3 | Apple iPhone SE | A1723 | iOS 13.5.1 |
| 4 | Apple iPhone 6 | A1586 | iOS 12.4.8 |
| 5 | Apple iPhone XS | A2097 | iOS 13.6.1 |
| 6 | Apple iPad 2 | MC769KN/A | iOS 9.3.5 |
| 7 | Samsung S4 Active | GT-I9295 | Android OS 5.0.1 |
| 8 | Samsung S5 Neo | SM-G903F | Android OS 6.0.1 |
| 9 | Samsung S6 | SM-G920F | Android OS 7 |
| 10 | Samsung S5 Neo | SM-G903F | Android OS 6.0.1 |

**Table A.1:** List of known devices

| # | **Brand** | **Model** | **Operating system** |
|---|---|---|---|
| 1 | OnePlus | 8 Pro | Android OS 10 |
| 2 | Google | Pixel 4 | Android OS 11 |
| 3 | Apple iPhone | XR | iOS 14.0.1 |
| 4 | Apple iPhone | 11 Pro | iOS 12.4.8 |
| 5 | Samsung | S20 | Android OS 10 |
| 6 | Apple iPhone | SE | iOS 13.6.1 |
| 7 | Apple iPhone | XR | iOS 14.0.1 |
| 8 | Apple iPhone | XR | iOS 13.7 |
| 9 | Samsung | S10e | Android OS 10 |
| 10 | Apple iPhone | 8 | iOS 14.1 |

**Table A.2:** List of unknown devices

Appendix B

# System architecture

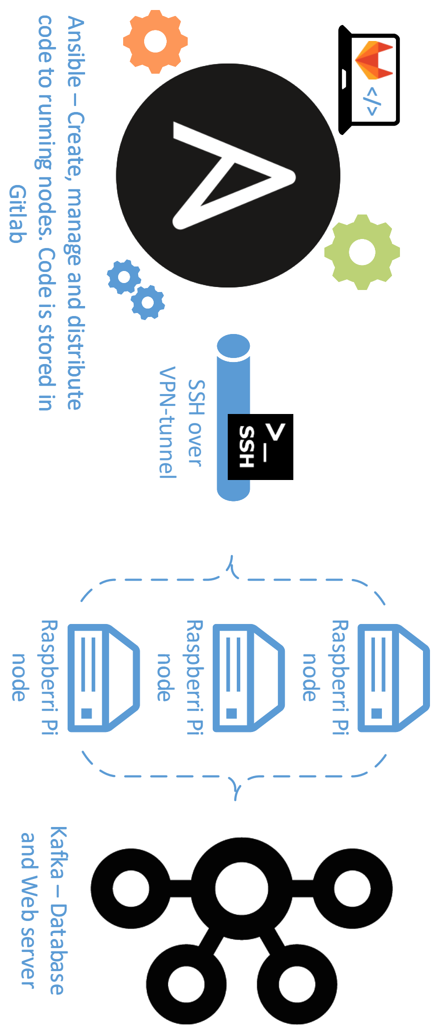

Ansible – Create, manage and distribute code to running nodes. Code is stored in Gitlab
</>
SSH over VPN-tunnel
SSH
Raspberri Pi node
Raspberri Pi node
Raspberri Pi node
Kafka – Database and Web server

**Figure B.1:** Architecture of the sensor network. Kafka was not used

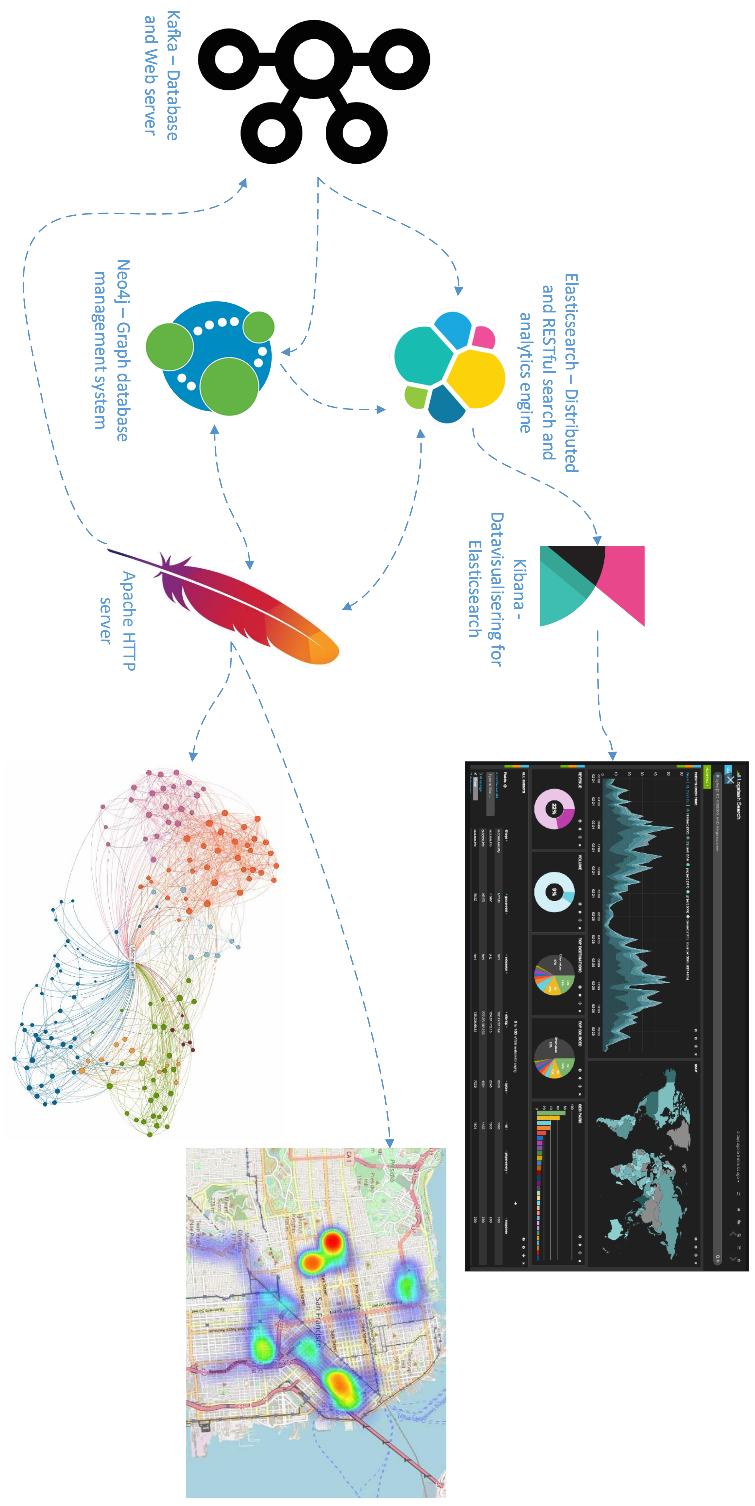

Kafka – Database and Web server
Neo4j – Graph database management system
Elasticsearch – Distributed and RESTful search and analytics engine
Apache HTTP server
Kibana - Datavisualisering for Elasticsearch

**Figure B.2:** Potential software which can be installed on a Kafka Webserver

# Appendix C

# Additional results

## C.1 Signal correlation

Scenario 2 (experiment 3-2, 3-3) and scenario 3 (6-2 and 6-3) were performed almost in the same way as scenario 1 (experiment 3-1 and 6-1). The difference was that scenario 2 used a different start-up procedure to see if Blue Hydra would reveal the Bluetooth Significant Address Part (SAP) faster. In scenario 3 the devices enabled their private Wi-Fi MAC address if the phone supported that. These differences should not have influenced the linking results.

### C.1.1 Linking devices in experiment 3-2

In experiment 3-2, data packets from nine out of ten devices were captured. The results, plotted in Figure C.1, show that MAC addresses were more rarely linked in top 1 compared to experiment 3-1. Algorithms using six sensors were more accurate than those using four sensors. Converting the signals from Bluetooth into Wi-Fi showed the most reliable results and performed with 78 % accuracy.

### C.1.2 Linking devices in experiment 3-3

Data packets from eight out of ten devices were in experiment 3-3 captured. The x-axis is updated to coincide with the number of devices when a MAC address was linked. The results from experiment 3-3 are plotted in Figure C.2. Here, the triangulation algorithm performs best with 86 % accuracy among the top 5, while the signal conversion into Wi-Fi signals (six sensors) appears more reliable with two devices linked in top 1.

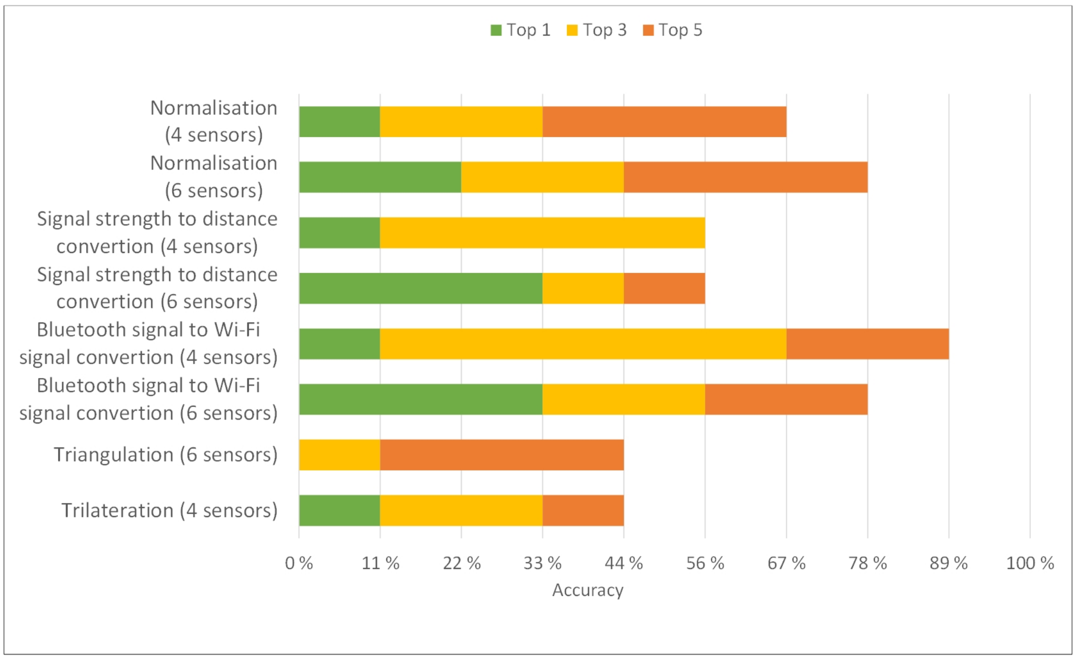


**Figure C.1:** Top K nearest neighbours from experiment 3-2

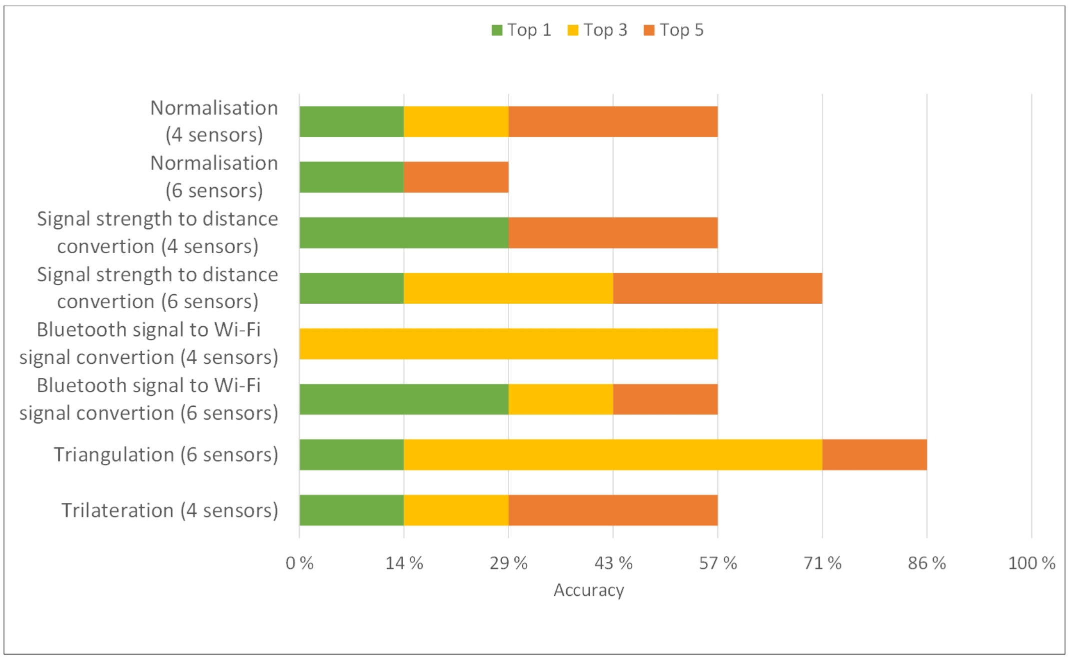


**Figure C.2:** Top K nearest neighbours from experiment 3-3

### C.1.3 Linking devices in experiment 6-2

Results from experiment 6-2 show poor accuracy of linking MAC addresses. The most reliable and best algorithm performed from this data is trilateration, where 50 % of the devices were linked among the top 5 devices.

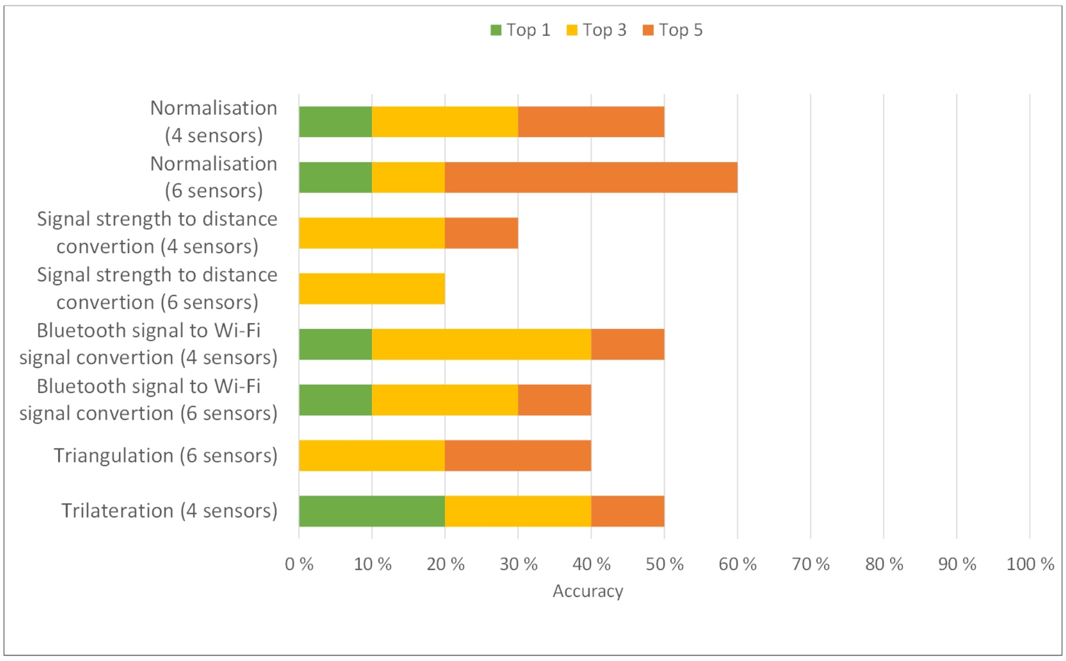


**Figure C.3:** Top K nearest neighbours from experiment 6-2

### C.1.4 Linking devices in experiment 6-3

In experiment 6-3, the method using normalisation with four sensors stands out as the algorithm that linked most MAC addresses in top 1 and top 3, with a total of 80 % in the top 5 devices.

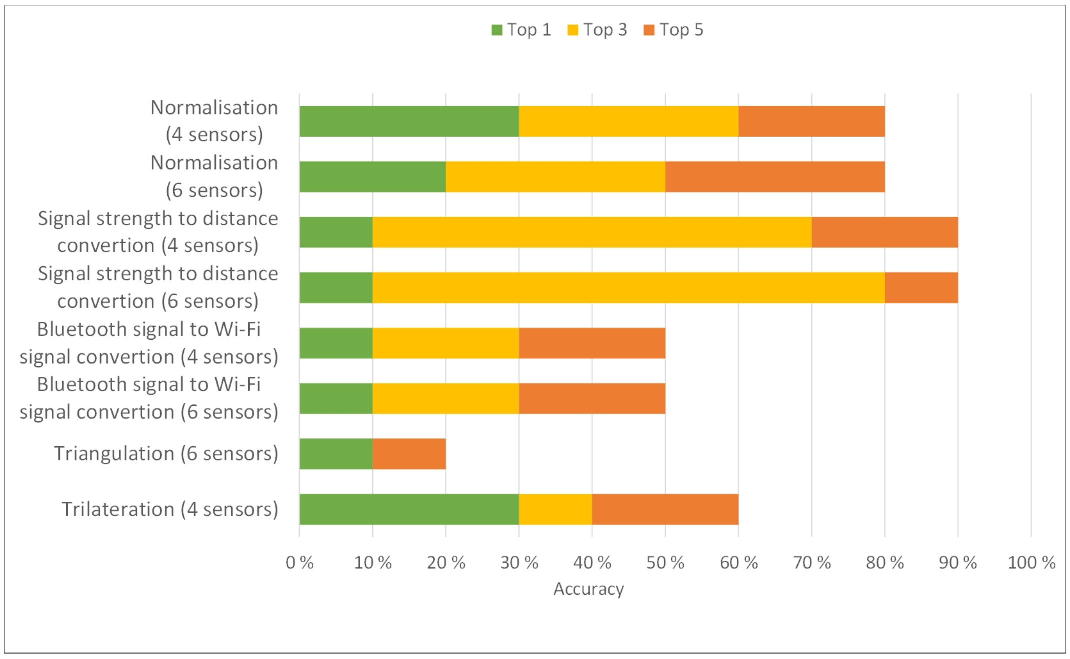


**Figure C.4:** Top K nearest neighbours from experiment 6-3

# Appendix D

# Matlab code

Due to the data collection size, it was necessary to manage the data in a smart way. Our solution generated two Matlab scripts in addition to a script for plotting the results. Script one created a connection to the SQL database and used SQL queries to import data into Matlab. In this script, the data was also thoroughly filtered and stored in smaller tables for further calculations. The second script used the stored tables in different linking and geolocation methods described in chapter 3. This appendix consists of script 1 and script 2.

- Step 1: Prepare data from SQL the database (filtering, calculate mean signal strength and convert the signal to distance)
- Step 2: Use prepared data in different methods (linking and geolocation)
- Step 3: Plot and visualise the results (script not included)

## Table of Contents

# Matlab Step 1 - Prepare data from SQL database

## Intro

@ Master thesis - Ensuring quality of covert police work with Wi-Fi and Bluetooth technology
@ Written in 2020/2021 by Atle Sørenssen, NTNU student

## MAC ADRESSES

```
% ____MAC Wi-Fi___________________________
% Removed in this version due to GDPR
% ____MAC Bluetooth_______________________
% Removed in this version due to GDPR
```

## Tables and variables

__Create tables______________________

```
sensor = char('s1','s2','s3','s4','s5','s6');
Results_static_signalstrength = strings(10,7);
Results_static_distance = strings(10,7);
Power_transmit = strings(10,7);
FSPLm= zeros(10,7);

% Static exp 3-1 - Calculated using device 10 as reference
% Wi-Fi specific - All data
PR_constant_wifi = [5;5.2499;    6.1701;    5.1694;   -0.2861;    5.1]; %
Constant to compensate for PR in each sensor
PT_powertransmit_wifi_known = 15; % Power Transmit Wifi - Reference number
PT_constant_wifi = [2.2;-4;2.4;1;3;0;-1;2.5;0;0]; %[3;-5;3;5;6;-2;-1.5;4;0;0];
% Bluetooth specific
PR_constant_bluetooth = [-1.5740; -2.2037;    2;   -0.6050;   -2.1730;   -
1.9316]; % Constant to compensate for PR in each sensor
PT_powertransmit_bluetooth_known = 8; % Power Transmit Bluetooth - Reference
number
```

```
 PT_constant_bluetooth = [4;-1.15;3;1;2;-3;5;-3;-0.3;0]; % Constant to
compensate for PT, number 10 is zero because its the reference device
```

## Conection to database

__Connect to database______________________

```
 dbfile = ('Atle_Loggfiler.db');
 conn = sqlite(dbfile);
```

## Filtering options

__Choose your config_____________________

```
     % A1 - Choose log file and MAC-list
 groupofdevices = 'unknown'; % known/unknown (if known, use device spesific
values obtained in experimet 1, else use mean values from experiment 1)
 mac_list = mac_unknown_6_1_6_2_wifi; % mac_known_bluetooth,
mac_unknown_ubertooth, mac_known_wifi, mac_unknown_wifi (dynamic),
mac_known_random_wifi, , mac_unknown_6_1_6_2_wifi, mac_unknown_random_6_3_wifi
 current_type = '"tcpdump_static"';  % Dynamic: tcpdump_dynamic,
ubertooth_dynamic. Static: ubertooth_static, tcpdump_static, l2ping, hcitool
 dynamic = 0; % 1/0 DYNAMIC/STATIC, choose between static (3-1,3-2,3-3,6-1,6-2)
or dynamic (4,5,7-1,7-2)

     % A2 - Choose options for device, experiment and filepath-value
 Devicenr = 5; % Select phone 1-10 (used only in dynamic experiments)
 mac_address = mac_list(Devicenr,:); % save current mac_address in variable
based on device number
 current_experiment = '"Eksperiment6-1"'; % (Eksperiment1 + meter, Eksperiment1-
2 + meter, Eksperiment3-1, Eksperiment3-2, Eksperiment3-3, Eksperiment4 + heat,
Eksperiment5, Eksperiment6-1, Eksperiment6-2, Eksperiment6-3, Eksperiment7-1,
Eksperiment7-2
 Filepath_value = '2020';  % ' or " eks. '2020', "_1m", "_2m", "_3m" ,..."_14m",
"_15m" eller "Heat1", "Heat2", "Heat3"

     % A3 - Choose a spesific time from/to
 time = 0;  % 1/0 - If yes, choose time below
 time_from = '21:49:32'; % From - Timestamp in GMT
 time_to = '21:52:13'; % To - Timestamp in GMT
 if time == 1 && contains(current_type,'ubertooth') % compensating for UTC+2
 time_from = datestr(datevec(time_from)+[0 0 0 2 0 0], 'HH:MM:SS'); % adding two
hours from GMT time
 time_to =datestr(datevec(time_to)+[0 0 0 2 0 0], 'HH:MM:SS'); % adding two
hours from GMT time
 end
```

```
    % B1 - Predefined - Choose settings related to signal strength filtering
 Filtrer_signalstyrke = 1; % 0/1, yes/no - If yes, choose a value
 if contains(current_type,'tcp') % Wi-Fi
 LessOrEqual_to = -15; % Less or equal to (strongest signal)
 GreaterOrEqual_to = -56; % Greater or equal to ((weakest signal), Wi-Fi limit:
-56 dBm
 else % Bluetooth
 LessOrEqual_to = -15; % Less or equal to (strongest signal)
 GreaterOrEqual_to = -73; % Greater or equal to (weakest signal), Bluetooth
limit: -73 dBm
 end
 Filter_signalstrength_in_percent = 0; % 1/0 = Yes/No - Filtering out the
highest and lowest values
 Percentfactor = 0.10;  % 0.1 = 10 %
 Filter_only_high = 0; % Filtering out the highest values (actually the lowest
values)
 Filter_only_low = 1; % Filtering out the lowest values (actually the highest
values)

    % B2 - Predefined - Choose values related to technology
 c_constant = 300*10^6;   % Speed of light in vacuum - Constant
 if contains(current_type,'tcp') % Wi-Fi frequency
 f_frequency = 2462*10^6; % 2462 (Wi-Fi, channel 11)
 else % Bluetooth frequency
 f_frequency = 2441*10^6; % 2441 (Bluetooth), Value in Mhz (Mean of 79 channels
from 2402 GHz to 2480 GHz - 2441 ((2480+2402)/2) = 2441 )
 end
 channel = 11; % Static channel for tcpdump (wifi), 2462 MHz.
```

## Pre check

__Pre check________________________

```
 tic % Start stopwatch
    % Check to choose between dBm and RSSI
    if contains(current_type,'hcitool') || contains(current_type,'l2ping')
        signal_strength_type = 'Signal_strength_RSSI'; % Signal type of hcitool
and l2ping
        time_format = 'Time';
    elseif contains(current_type, 'tcpdump_dynamic')
         current_type = '"tcpdump_whitelisted_combined_seconds_distance"'; %
Correct name of table in SQL
        signal_strength_type = 'Signal_strength_dBm_2_Average_Limit56';
        time_format = 'Time_2';
    elseif contains(current_type, 'ubertooth_dynamic')
```

```
        current_type = '"ubertooth_whitelisted_combined_seconds_distance"'; %
Correct name of table in SQL
        signal_strength_type = 'Signal_strength_dBm_Average_Limit73'; %
Signal_strength_dBm_Average_Limit73
        time_format = 'Time_2';
    elseif contains(current_type,'ubertooth_static')
        current_type = '"ubertooth_whitelisted_distance"'; % Correct name of
table in SQL
        signal_strength_type = 'Signal_strength_dBm'; % Signal received with
ubertooth hardware (only one antenna)
        time_format = 'Time';
    elseif contains(current_type,'tcpdump_static')
        current_type = '"tcpdump_whitelisted_distance"'; % New table in SQL
where recevied signals (in antenna 0) equal to zero have been removed. This
gives more realiable mean values signals received with antenna 1
        signal_strength_type = 'Signal_strength_dBm_2'; % Signal value (antenna
1) from wireshark that occurs to be more accurate
        time_format = 'Time';
    end

 % ____Pre data_________________________
 if contains(current_type,'tcpdump_whitelisted_combined_seconds_distance')
        pre_sqlquery = ['SELECT
',time_format,',Eksperiment,Source,round(',signal_strength_type,',4),Sensor,File
path,Channel,Combined,Sum_signalstrength FROM ',current_type,' WHERE
"Eksperiment" LIKE ',current_experiment,' ESCAPE ''\'' '];
        pre_data = fetch(conn,pre_sqlquery);
        pre_data = cell2table(pre_data,...
                'VariableNames',{'Time' 'Eksperiment' 'MAC Wi-Fi'
'Signal_strength_dBm_2_Average_Limit56' 'Sensor' 'Filepath' 'Channel' 'Combined'
'Sum signalstrength'});
 elseif contains(current_type,'tcpdump_whitelisted_distance')
        pre_sqlquery = ['SELECT
',time_format,',Eksperiment,Source,round(',signal_strength_type,',4),Sensor,File
path,Channel FROM ',current_type,' WHERE "Eksperiment" LIKE
',current_experiment,' ESCAPE ''\'' '];
        pre_data = fetch(conn,pre_sqlquery);
        pre_data = cell2table(pre_data,...
                'VariableNames',{'Time' 'Eksperiment' 'MAC Wi-Fi'
'Signal_strength_dBm_2_Average_Limit56' 'Sensor' 'Filepath' 'Channel'});
 elseif (contains(current_type, 'hcitool')) && (contains(current_experiment, '1-
2'))
        pre_sqlquery = ['SELECT
',time_format,',Eksperiment,Source,round(',signal_strength_type,',4),Sensor,File
name FROM ',current_type,' WHERE "Eksperiment" LIKE ',current_experiment,' '];
        pre_data = fetch(conn,pre_sqlquery);
```

```
         pre_data = cell2table(pre_data,...
                'VariableNames',{'Time' 'Eksperiment' 'MAC Bluetooth'
'Signalstyrke RSSI' 'Sensor' 'Filename'});
 elseif (contains(current_type, 'l2ping')) || (contains(current_type,
'hcitool'))
         pre_sqlquery = ['SELECT
',time_format,',Eksperiment,Source,round(',signal_strength_type,',4),Sensor,File
path FROM ',current_type,' WHERE "Eksperiment" == ',current_experiment,' '];
         pre_data = fetch(conn,pre_sqlquery);
         pre_data = cell2table(pre_data,...
                'VariableNames',{'Time' 'Eksperiment' 'MAC Bluetooth'
'Signalstyrke RSSI' 'Sensor' 'Filepath'});
 elseif contains(current_type,
'ubertooth_whitelisted_combined_seconds_distance')
         pre_sqlquery = ['SELECT
',time_format,',Eksperiment,Source,round(',signal_strength_type,',4),Sensor,File
path,Combined,Sum_signalstrength FROM ',current_type,' WHERE "Eksperiment" LIKE
',current_experiment,' '];
         pre_data = fetch(conn,pre_sqlquery);
         pre_data = cell2table(pre_data,...
                'VariableNames',{'Time' 'Eksperiment' 'MAC Bluetooth'
'Signal_strength_dBm_Average_Limit67' 'Sensor' 'Filepath' 'Combined' 'Sum
signalstrength'});  % Signal_strength_dBm_Average_Limit67
 elseif contains(current_type, 'ubertooth_whitelisted_distance')
         pre_sqlquery = ['SELECT
',time_format,',Eksperiment,Source,round(',signal_strength_type,',4),Sensor,File
path FROM ',current_type,' WHERE "Eksperiment" LIKE ',current_experiment,' '];
         pre_data = fetch(conn,pre_sqlquery);
         pre_data = cell2table(pre_data,...
                'VariableNames',{'Time' 'Eksperiment' 'MAC Bluetooth'
'Signal_strength_dBm_Average' 'Sensor' 'Filepath'});  %
Signal_strength_dBm_Average_Limit67
 end

 Rows_pre = size(pre_data{:,4});  % Saves number of rows related to experiment
 pre_data = sortrows(pre_data,[1],{'ascend'});
```

## Filtering + Friis calculations

```
 if dynamic == 1 % Following one MAC/UAP address through the rest of the code
```

## DYNAMIC

__Ready to FILTER__________

```
     % Filtering down to sensors, then calculate or put value in new table
```

```
    for s=1:6
        current_data = pre_data; % Sets current data to the data obtained from
SQL (pre_data)
        current_data =
current_data(contains(current_data.(3),mac_address(1,1:8)),:); % Filtering the
spesific MAC/UAP address
        current_data =
current_data(contains(current_data.(6),Filepath_value),:); % Filtering filename
        if contains(current_type,'tcp') % Filtering channel 11 in table tcpdump
           current_data = current_data(current_data.(7) == 11,:); % Filtering
out other channels than 11 in tcpdump log
        end
        if Filtrer_signalstyrke == 1
           current_data = current_data(current_data.(4) <= LessOrEqual_to,:); %
Filtering on the strongest value
           current_data = current_data(current_data.(4) >=
GreaterOrEqual_to,:); % Filtering on the weakest value
        end
        if time == 1
           if height(current_data) >=1 % Check if there are any data between
the timestamps
               current_data = current_data(datenum(current_data.Time) >=
datenum(time_from),:);
               current_data = current_data(datenum(current_data.Time) <=
datenum(time_to),:);
             else
                continue % Passes control to the next iteration of s (sensor) if
no data
           end
        end
        if contains(current_type,'ubertooth') % compensating for UTC+2 in
Ubertooth data
             current_time_ubertooth = current_data.Time;
             new_time_ubertooth = datestr(datevec(current_time_ubertooth)-[0 0 0
2 0 0], 'HH:MM:SS'); % adjusting into UTC+0
             current_data.Time = new_time_ubertooth; % Update current data with
new time
        end
        if height(current_data) == 0 % Final check to see if there are any data
left before calculation
             if (s == 6 && height(current_data) == 0)
                 msg = 'No data from this device.' % Message will occur in
command line if no data is detected
                 return;
             end
             continue % Passes control to the next iteration of s (sensor) if no
data
         end
        if s ==1 % Setting timestamps in current data in column 1 - do it once
when s=1
          current_data_timestamps =
sortrows(string(current_data{:,1}),{'ascend'}); % Converting to string and
sorting timestamps from start to end
          current_data_timestamps_unique = unique(current_data_timestamps(:,1));
          sizecurrent_data = size(current_data_timestamps_unique(:,1));
          Results_dynamic_signalstrength = strings(sizecurrent_data(1,1),7); %
preallocating
          Results_dynamic_distance = strings(sizecurrent_data(1,1),7); %
preallocating
          Results_dynamic_distance(1:sizecurrent_data,1) =
unique(current_data_timestamps(:,1));
          Results_dynamic_signalstrength(1:sizecurrent_data,1) =
unique(current_data_timestamps(:,1));
        end
         current_data =
current_data(contains(current_data.(5),sensor(s,1:2)),:); % Filtering on current
sensor
 % ______End of filtering DYNAMIC____________
 % ______Calculation begins DYNAMIC__________
             % For every timestamp put signalstrength in
Results_dynamic_signalstrength or calcualte distance
             for t=1:size(current_data(:,1)) % Do this loop for the length of
current data (different amount of data on each sensor)

                if contains(current_type,'tcp') % Wi-Fi
                    PR_powerreceiver = PR_constant_wifi(s) +
(current_data{t,4}); % Save current receiving signal value within that second as
PR_powerreceiver and add calibrated value for sensor
                else % Bluetooth
                    PR_powerreceiver = PR_constant_bluetooth(s) +
(current_data{t,4}); % Save current receiving signal value within that second as
PR_powerreceiver and add calibrated value for sensor
                end
                % Find best PT_powertransmit based on info from Experiment 1
                mac_address_numberinlist =
find((all(ismember(mac_list,mac_address),2))==1); % To link MAC address to
PT_powertransmit
                if contains(current_type,'tcp')
                    if contains(groupofdevices,'unknown')
                         PT_powertransmit_current = PT_powertransmit_wifi_known;
% Same default value for known/unknown device (15 dbm)
```

```
                      else
                           PT_powertransmit_current = PT_powertransmit_wifi_known
+ PT_constant_wifi(mac_address_numberinlist); % Default output power (15dbm) +
individual constant
                      end
                  else
                      if contains(groupofdevices,'unknown')
                           PT_powertransmit_current =
PT_powertransmit_bluetooth_known; % Same default value for known/unknown device
(8 dbm)
                      else
                          PT_powertransmit_current =
PT_powertransmit_bluetooth_known +
PT_constant_bluetooth(mac_address_numberinlist); % Default output power (8dbm) +
individual constant
                      end
                  end
                  FSPLm=double(PT_powertransmit_current-PR_powerreceiver); %
signal path loss
                  timestamp_numberinlist =
find(all(ismember(current_data_timestamps_unique,current_data{t,1}),2)); % Find
matching timestamp in current data where timestamp is unique
                  Results_dynamic_distance(timestamp_numberinlist,s+1) =
10^((FSPLm-(20*log10(f_frequency))-(20*log10(4*pi/c_constant)))/20); % Using
Friis equation to calculate signal to distance
                  Results_dynamic_signalstrength(timestamp_numberinlist,s+1) =
double(current_data{t,4}); % Putting signalstrength in table
                  t=t+1; % Increment row of timestamp with 1
              end
         s=s+1; % Increment sensorID with 1
     end
      % Setting variable names to the table and converts to table
      Results_dynamic_signalstrength =
array2table(Results_dynamic_signalstrength,...
            'VariableNames',{'Timestamp' 's1' 's2' 's3' 's4' 's5' 's6'});
      Results_dynamic_distance = array2table(Results_dynamic_distance,...
            'VariableNames',{'Timestamp' 's1' 's2' 's3' 's4' 's5' 's6'});
     if (contains(current_type,'tcpdump'))
         Results_dynamic_wifi_signalstrength = Results_dynamic_signalstrength;
         Results_dynamic_wifi_distance = Results_dynamic_distance;
         fname = sprintf('Output/@Dynamic/R_dyn_wifi_dis_exp%s_device-%d.mat',
current_experiment, Devicenr);
         save (fname, 'Results_dynamic_wifi_distance'); % Saves the value of
Results_.. in the file Results_...mat. This will be later used in
Studie_Master_Step2.m
```

```
         fname = sprintf('Output/@Dynamic/R_dyn_wifi_sig_exp%s_device-%d.mat', 
current_experiment, Devicenr);
         save (fname, 'Results_dynamic_wifi_signalstrength');
     else
         Results_dynamic_bluetooth_signalstrength = 
Results_dynamic_signalstrength;
         Results_dynamic_bluetooth_distance = Results_dynamic_distance;
         fname = sprintf('Output/@Dynamic/R_dyn_blue_dis_exp%s_device-%d.mat', 
current_experiment, Devicenr);
         save (fname, 'Results_dynamic_bluetooth_distance'); % Saves the value 
of Results_.. in the file Results_...mat. This will be later used in 
Studie_Master_Step2.m
         fname = sprintf('Output/@Dynamic/R_dyn_blue_sig_exp%s_device-%d.mat', 
current_experiment, Devicenr);
         save (fname, 'Results_dynamic_bluetooth_signalstrength');
     end
 else
```

## STATIC

```
 Totalrows = 1; % Manual counter to compare to SQLdatabase
 % ____Ready to FILTER STATIC experiment______________
 for m=1:size(mac_list(:,1)) % Do this loop for the length of mac_list
         if (contains(current_experiment,'t1') || 
contains(current_experiment,'1-2')) && (contains(current_type,'tcp') || 
contains(current_type,'ubertooth'))
             s=1;    % Number of sensors in experiment 1 is one
         elseif (contains(current_experiment,'t1') || 
contains(current_experiment,'1-2')) && (contains(current_type,'l2ping') || 
contains(current_type,'hcitool'))
             s=2;
         else
             s=1:6; % Number of sensors in the other experiments are six
         end
             for s=s
                current_data = pre_data;
                    if contains(current_type,'tcp') % Filtering channel 11 in 
table tcpdump
                       current_data = current_data(current_data.(7) == 11,:); % 
Filtering out other channels than 11 in tcpdump log
                    end
                current_data = 
current_data(contains(current_data.(3),mac_list(m,1:8)),:); % Filtering on MAC
                current_data = 
current_data(contains(current_data.(5),sensor(s,1:2)),:); % Filtering on sensor
```

```
                current_data =
current_data(contains(current_data.(6),Filepath_value),:); % Filtering on
filename
                    if time == 1
                        if height(current_data) >=1 % Check if there are any
data between the timestamps
                            current_data =
current_data(datenum(current_data.Time) >= datenum(time_from),:);
                            current_data =
current_data(datenum(current_data.Time) <= datenum(time_to),:);
                        else
                            continue % Passes control to the next iteration of s
(sensor) if no data
                        end
                    end
                    if height(current_data) == 0 % Final check to see if there
are any data left before calculation
                         continue % Passes control to the next iteration of s
(sensor) if no data
                    end
                    if Filtrer_signalstyrke == 1
                        if height(current_data) >=1 % Check if there are any
data left
                        current_data = current_data(current_data.(4) <=
LessOrEqual_to,:); % Filtering on the strongest value
                        current_data = current_data(current_data.(4) >=
GreaterOrEqual_to,:); % Filtering on the weakest value
                        else
                            continue % Passes control to the next iteration of s
(sensor) if no data
                        end
                    end
                Rows = height(current_data); % Check current rows in current
data
                    if Rows >=1 % Quit if there are no data
                             if Rows > 5
                                     current_data =
sortrows(current_data,[4],{'descend'});  % Sort with highest value
                                         % Filtering received signals,
10 % of the HIGHEST and LOWEST signals from every MAC address related to each
sensor
                                     if Filter_signalstrength_in_percent ==
1 && Filter_only_low == 0 && Filter_only_high == 0
                                         x_length_current_data =
round(height(current_data)*Percentfactor);  % Choosing to remove X % of values
```

```
                                             length_current_data =
height(current_data);
                                             y_length_current_data =
length_current_data - x_length_current_data;
                                             current_data =
current_data(1:y_length_current_data,:);
                                             current_data =
current_data(x_length_current_data:y_length_current_data,:);

                                                 % Filtering away X % of the
HIGHER signal values
                                         elseif
(Filter_signalstrength_in_percent == 1 && Filter_only_low == 1)
                                             x_length_current_data =
round(height(current_data)*Percentfactor);  % Choosing to remove X % of values
                                             current_data =
current_data(x_length_current_data:end,:);

                                                  % Filtering away X % of the
LOWER signal values
                                         elseif
(Filter_signalstrength_in_percent == 1 && Filter_only_high == 1)
                                             x_length_current_data =
round(height(current_data)*Percentfactor);  % Choosing to remove X % of values
                                             length_current_data =
height(current_data);
                                             y_length_current_data =
length_current_data - x_length_current_data;
                                             current_data =
current_data(1:y_length_current_data,:);
                                         else
                                         end
                                 else
                                 end
                                 Totalrows = Totalrows+Rows;
                        % ______End of filtering STATIC________________
                        % ______Calculation begins STATIC______________
                                if height(current_data) > 0 % Checks if there
exists rows after filtering
                                            if contains(current_type,'tcp') %
Wi-Fi

Results_static_signalstrength(m,s+1)= PR_constant_wifi(s) +
mean(current_data{:,4}); % Calibrate each sensor. Calculate mean value of
current data.
```

```
                                                PR_powerreceiver =
PR_constant_wifi(s) + mean(current_data{:,4}); % Save current receiving signal-
value in own variable after calibration.
                                            else % Bluetooth

Results_static_signalstrength(m,s+1)= PR_constant_bluetooth(s) +
mean(current_data{:,4}); % Calibrate each sensor. Calculate mean value of
current data.
                                                PR_powerreceiver =
PR_constant_bluetooth(s) + mean(current_data{:,4}); % Save current receiving
signal-value in own variable after calibration.
                                            end
                                end
                                        % Calibrate PT_powertransmit based on
info from Experiment 3-1
                                        if contains(current_type,'tcp')
                                            if
contains(groupofdevices,'unknown')
                                                 PT_powertransmit_current =
PT_powertransmit_wifi_known; % + PT_constant_wifi(m);
                                            else
                                                 PT_powertransmit_current =
PT_powertransmit_wifi_known + PT_constant_wifi(m); % PT_powertransmit_wifi_known
= 15; % Mean value from PT in experiment 3-1 (Wi-Fi)
                                            end
                                        else
                                            if
contains(groupofdevices,'unknown')
                                                 PT_powertransmit_current =
PT_powertransmit_bluetooth_known; % + PT_constant_bluetooth(m); % Device
spesific obtained in experiment 3-1
                                            else
                                                PT_powertransmit_current =
PT_powertransmit_bluetooth_known + PT_constant_bluetooth(m); %
PT_powertransmit_bluetooth_known = 8; % Mean value from PT in experiment 3-1
(Bluetooth)
                                            end
                                        end

FSPLm(m,s+1)=double(PT_powertransmit_current-PR_powerreceiver); % Calculate free
space path loss
                                         if contains(groupofdevices,'known')
                                            if contains(current_type,'tcp') %
Wi-fi

Results_static_signalstrength(m,s+1) = PT_powertransmit_wifi_known -
```

```
FSPLm(m,s+1);   % Changing PR if PT is individually calibrated (not changing for
unknown devices..)
                                             else % Bluetooth

Results_static_signalstrength(m,s+1) = PT_powertransmit_bluetooth_known -
FSPLm(m,s+1);   % Changing PR if PT is individually calibrated (not changing for
unknown devices..)
                                             end
                                         else  % do not change unkown devices
                                         end

                                        if (contains(current_type,'l2ping') ||
contains(current_type,'hcitool'))
                                             Results_static_distance(m,s+1) =
(10^((FSPLm(m,s+1)-(20*log10(f_frequency))-(20*log10(4*pi/c_constant)))/20));
% Using Friis equation to calculate signal to distance
                                        else
                                             Results_static_distance(m,s+1) =
10^((FSPLm(m,s+1)-(20*log10(f_frequency))-(20*log10(4*pi/c_constant)))/20);
% Using Friis equation to calculate signal to distance
                                        end

Results_static_signalstrength(m,1)=current_data{1,3}; % Setting current MAC
address in table

Results_static_distance(m,1)=current_data{1,3}; % Setting current MAC address in
table
                                     Power_transmit(m,1)=current_data{1,3}; %
Setting current MAC address in table

                                     if (m==1 || m==2) &&
(contains(current_experiment,'"3-3"'))  % Ignore known device #1, #2 in exp 3-3
- See error matrix in analysis chapter

Results_static_signalstrength(m,s+1)='omitnan';

Results_static_distance(m,s+1)='omitnan';
                                     end
                                     if m==9 &&
(contains(current_experiment,'"3-2"') || contains(current_experiment,'"3-3"')) %
Ignore known device #9 in exp 3-2 and 3-3 - See error matrix in analysis chapter

Results_static_signalstrength(m,s+1)='omitnan';

Results_static_distance(m,s+1)='omitnan';
                                     end
```

```
                        % ______End of calculations__________________
                    else
                        break
                    end
                    s=s+1; % Increment sensorID with 1
             end
             m=m+1; % Increment MAC address with 1
 end
        % Setting variable names to the table and converts to table
      Results_static_signalstrength =
array2table(Results_static_signalstrength,...
            'VariableNames',{'MAC' 's1' 's2' 's3' 's4' 's5' 's6'});
      Results_static_distance = array2table(Results_static_distance,...
            'VariableNames',{'MAC' 's1' 's2' 's3' 's4' 's5' 's6'});
     if (contains(current_type,'tcpdump'))
     Results_static_wifi_signalstrength = Results_static_signalstrength;
     Results_static_wifi_distance = Results_static_distance;
     % Saves the value of Results_.. in the file Results_...mat. This will be
later used in Studie_Master_Step2.m
     save Output/@Static/Results_static_wifi_distance.mat
Results_static_wifi_distance;
     save Output/@Static/Results_static_wifi_signalstrength.mat
Results_static_wifi_signalstrength;
     else
     Results_static_bluetooth_signalstrength = Results_static_signalstrength;
     Results_static_bluetooth_distance = Results_static_distance;
     % Saves the value of Results_.. in the file Results_...mat. This will be
later used in Studie_Master_Step2.m
     save Output/@Static/Results_static_bluetooth_distance.mat
Results_static_bluetooth_distance;
     save Output/@Static/Results_static_bluetooth_signalstrength.mat
Results_static_bluetooth_signalstrength ;
     end
 end
 clear
 toc % Stop stopwatch
```

## Notes

__Mathematics_______________

```
     %d_distance = 10^((FSPLm_bluetooth-(20*log10(f_frequency_bluetooth))-
(20*log10(4*pi/c_constant)))/20)

 % ____Filtering tips____________
```

```
    % current_data = sortrows(current_data,[1],{'ascend'});
    % current_data(:,1:5)
    % sortrows(current_data(:,1:5),[4],{'descend'})
    % sortrows(pre_data(:,1:5),[4],{'descend'})
    % t = {'12:34:56'};
    % dnt = datenum(current_data.Time)
    % str2double(string(Resultat_distanse{1,2}))
```

## Table of Contents

## Matlab Step 2 - Use prepared data from step 1 in linking and geolocation methods

### Variables

```
 xy = [0 0; 11.195 0; 0 20.80; 11.195 20.80]'; % Static coordinates of sensors
to be used in trilateration
 sensors_triangulation = [0 0; 11.195 0; 0 10.4; 11.195 10.40; 0 20.80; 11.195
20.80]'; % Static position of sensors (coordinates)
 locations = [7.7606 3.7000; 2.6216 3.7120; 2.6738 9.0360; 5.6953 7.3921; 8.6681
9.4556; 10.1249 14.5731; 2.4515 12.2649; 7.1954 16.3039; 3.0962 19.3244; 6.8989
10.7836]; % static locations where the decives were located during static
experiments
 base_color = [0 0 0;0 0.4470 0.7410;0.8500 0.3250 0.0980; 0.4290 0.4940 0.4250;
0.4940 0.1840 0.5560;0.4660 0.6740 0.1880;0.3010 0.7450 0.9330;0.6350 0.0780
0.1840;1.0000 1.0000 0;1.0000 0.6633 0]; % Define colormap
```

### Input values

```
 static = 1; % 1/0, Static/Dynamic (experiment type)
 device = 10; % Dynamic only - choose device from step 1 to be loaded
 experiment = '4'; % Dynamic only - choose experiment
 groupofdevices = 1; % Known/Unknown 1/0 (Only in static.. when 0 i selected, a
mean value from the known devices are used)
 method = 1; % 1/2/3 - Linking/Triangulation/Trilateration
```

### Loading tables

```
 if static == 1
 % Static files
 load Output/@Static/Results_static_wifi_distance; % Import of static results
from step 1
 Results_static_wifi_distance =
str2double(string(Results_static_wifi_distance{:,2:7})); % Converting only the
results to double
 load Output/@Static/Results_static_wifi_signalstrength;
 Results_static_wifi_signalstrength =
str2double(string(Results_static_wifi_signalstrength{:,2:7}));
 load Output/@Static/Results_static_bluetooth_distance;
 Results_static_bluetooth_distance =
str2double(string(Results_static_bluetooth_distance{:,2:7}));
 load Output/@Static/Results_static_bluetooth_signalstrength;
 Results_static_bluetooth_signalstrength =
str2double(string(Results_static_bluetooth_signalstrength{:,2:7}));
 else
 % Dynamic files
 roothpath = 'Output/@Dynamic/';
 exp_devices=sprintf('*%s"_device-%d', experiment,device);
 findfiles = struct2cell(dir(fullfile(roothpath,[exp_devices,'.mat'])))';
 files = string(findfiles(:,1));
     if length(files(:,1)) < 4 % Check for both Wifi and Bluetooth files
         msg = 'Skipping device - Only bluetooth or Wifi'
         return;
     end
   for f=1:length(files)
     current_filename = files(f);
     files_with_roothpath = sprintf('%s%s',roothpath,files(f));
     if (contains(files(f),'wifi') && contains(files(f),'dis'))
         load (files_with_roothpath);
         Fulltable_dynamic_wifi_distance = Results_dynamic_wifi_distance;  %
Allocating this variable with full table including timestamps
         Results_dynamic_wifi_distance =
str2double(string(Results_dynamic_wifi_distance{:,2:7})); % Converting only the
results to double
     elseif (contains(files(f),'blue') && contains(files(f),'dis'))
        load (files_with_roothpath);
        Fulltable_dynamic_bluetooth_distance =
Results_dynamic_bluetooth_distance;
        Results_dynamic_bluetooth_distance =
str2double(string(Results_dynamic_bluetooth_distance{:,2:7}));
     end
   end
 end
```

## Experiment type

```
 if static == 1 % Choosing static results based on input values
```

## STATIC

Methods - Linking / Triangulation / Multilateration

```
     if method == 1 % Linking Wi-Fi and Bluetooth signals + Top K-values
```

### 1 Normalization + Euclidean + Top K-values - 4/6 sensors

```
     % Variables needed in normalization
     Results_bluetooth_signalstrength_double_norm=zeros(10,7);
     Results_wifi_signalstrength_double_norm=zeros(10,7);

     % Normalizing both the wi-fi and bluetooth data
     for r=1:10 % loop through every device in both bluetooth and wi-fi data
     Results_bluetooth_signalstrength_double_norm(r,2:7) =
(Results_static_bluetooth_signalstrength(r,:) -
min(Results_static_bluetooth_signalstrength(r,:))) /
(max(Results_static_bluetooth_signalstrength(r,:)) -
min(Results_static_bluetooth_signalstrength(r,:)));
     Results_bluetooth_signalstrength_double_norm(r,1) = r; % Setting current
device in first column
     Results_wifi_signalstrength_double_norm(r,2:7) =
(Results_static_wifi_signalstrength(r,:) -
min(Results_static_wifi_signalstrength(r,:))) /
(max(Results_static_wifi_signalstrength(r,:)) -
min(Results_static_wifi_signalstrength(r,:)));
     Results_wifi_signalstrength_double_norm(r,1) = r; % Setting current device
in first column


     % result(k,:) = (x(k,:) - min(x(k,:))) ./ (max(x(k,:)) - min(x(k,:)));
             % All values from sensor 1-6 are normalized with one command
             % x: data from six sensors in a table (10x6)
             % k: row in data
             % result: i(th) normalized data
             % xmin: lowest value from the 6 sensors
             % xmax: highest value from the 6 sensors
     end
     % Euclideaon distance - Testing each Wi-Fi MAC address against each
Bluetooth MAC address
 length=0;
     for wr=1:10 % loop through the normalized wifi data - wr (wifi-row)
         for br=1:10  % loop through the normalized bluetooth data - br
(bluetooth-row)
```

```
             length = length+1;
             % Calculation based on 6 sensors (1,2,3,4,5,6) - Euclidean distance
between two points (in this case between the Bluetooth data and Wi-Fi data)
             Results_signalstrength_double_norm_euclideon_6sensros(length,1) =
wr; % Saves Wi-Fi device number in column 1
             Results_signalstrength_double_norm_euclideon_6sensros(length,2) =
br; % Saves Bluetooth device number in column 2
             Results_signalstrength_double_norm_euclideon_6sensros(length,3) =
sqrt(((Results_wifi_signalstrength_double_norm(wr,1+1)-
Results_bluetooth_signalstrength_double_norm(br,1+1))^2)+((Results_wifi_signalst
rength_double_norm(wr,2+1)-
Results_bluetooth_signalstrength_double_norm(br,2+1))^2)+((Results_wifi_signalst
rength_double_norm(wr,3+1)-
Results_bluetooth_signalstrength_double_norm(br,3+1))^2)+((Results_wifi_signalst
rength_double_norm(wr,4+1)-
Results_bluetooth_signalstrength_double_norm(br,4+1))^2)+((Results_wifi_signalst
rength_double_norm(wr,5+1)-
Results_bluetooth_signalstrength_double_norm(br,5+1))^2)+((Results_wifi_signalst
rength_double_norm(wr,6+1)-
Results_bluetooth_signalstrength_double_norm(br,6+1))^2));
             % Calculation based on 4 sensors (1,2,5,6) - Euclidean distance
between two points (in this case between the Bluetooth and Wi-Fi values)
             Results_signalstrength_double_norm_euclideon_4sensros(length,1) =
wr; % Saves Wi-Fi device number in column 1
             Results_signalstrength_double_norm_euclideon_4sensros(length,2) =
br; % Saves Bluetooth device number in column 2
             Results_signalstrength_double_norm_euclideon_4sensros(length,3) =
sqrt(((Results_wifi_signalstrength_double_norm(wr,1+1)-
Results_bluetooth_signalstrength_double_norm(br,1+1))^2)+((Results_wifi_signalst
rength_double_norm(wr,2+1)-
Results_bluetooth_signalstrength_double_norm(br,2+1))^2)+((Results_wifi_signalst
rength_double_norm(wr,5+1)-
Results_bluetooth_signalstrength_double_norm(br,5+1))^2)+((Results_wifi_signalst
rength_double_norm(wr,6+1)-
Results_bluetooth_signalstrength_double_norm(br,6+1))^2));

             % Format: d(w, b) = sqrt((w(s1)-b(s1))^2+(w(s2)-b(s2))^2);
                 % d(w, b): is close to 0 if the two rows are very similar and
became greater if the lines are different
                 % w: The normalized wifi value
                 % b: The normalized bluetooth value
                 % s: sensor-1, sensor-2,... sensor-6
         end
     end
     % Top K-values
     data_device_from = 1;
     data_device_to = 10;
     for data_device = 1:10
         % 6 sensors
         current_data =
sortrows(Results_signalstrength_double_norm_euclideon_6sensros(data_device_from:
data_device_to,:),[3],{'ascend'}); % For every device in
"Results_signalstrength_double_norm_euclideon", sort the values in ascending
order

TopK_6sensors_signalstrength_double_norm_euclideon(data_device_from:data_device_
to,:) = current_data; % Final values to look through when plotting into graph
         % 4 sensors
         current_data =
sortrows(Results_signalstrength_double_norm_euclideon_4sensros(data_device_from:
data_device_to,:),[3],{'ascend'}); % For every device in
"Results_signalstrength_double_norm_euclideon", sort the values in ascending
order

TopK_4sensors_signalstrength_double_norm_euclideon(data_device_from:data_device_
to,:) = current_data; % Final values to look through when plotting into graph
         data_device_from = data_device_from + 10; % sliding windows - next 10
devices
         data_device_to = data_device_to + 10; % sliding windows - next 10
devices
     end
```

## 2 Bluetooth/Wi-Fi signal to distance + Euclidean + Top K-values - 4/6 sensors

```
    % Euclideaon distance - Testing each Wi-Fi MAC address against each
Bluetooth MAC address using distance as input (calculated in
"Study_Master_step1_static.m" with Friis equation)
 length=0;
 for wr=1:10 % loop through the converted distance wifi data - wr (wifi-row)
     for br=1:10  % loop through the converted distance bluetooth data - br
(bluetooth-row)
         length = length+1;
         % Calculation based on 6 sensors (1,2,3,4,5,6) - Euclidean distance
between two points (in this case between the Bluetooth and Wi-Fi values)
         Results_distance_double_signaltodistance_euclideon_6sensors(length,1) =
wr; % Saves Wi-Fi device number in column 1
         Results_distance_double_signaltodistance_euclideon_6sensors(length,2) =
br; % Saves Bluetooth device number in column 2
         Results_distance_double_signaltodistance_euclideon_6sensors(length,3) =
sqrt((Results_static_wifi_distance(wr,1)-
Results_static_bluetooth_distance(br,1))^2+(Results_static_wifi_distance(wr,2)-
Results_static_bluetooth_distance(br,2))^2+(Results_static_wifi_distance(wr,3)-
Results_static_bluetooth_distance(br,3))^2+(Results_static_wifi_distance(wr,4)-
Results_static_bluetooth_distance(br,4))^2+(Results_static_wifi_distance(wr,5)-
Results_static_bluetooth_distance(br,5))^2+(Results_static_wifi_distance(wr,6)-
Results_static_bluetooth_distance(br,6))^2);
         % Calculation based on 4 sensors (1,2,5,6) - Euclidean distance between
two points (in this case between the Bluetooth and Wi-Fi values)
         Results_distance_double_signaltodistance_euclideon_4sensors(length,1) =
wr; % Saves Wi-Fi device number in column 1
         Results_distance_double_signaltodistance_euclideon_4sensors(length,2) =
br; % Saves Bluetooth device number in column 2
         Results_distance_double_signaltodistance_euclideon_4sensors(length,3) =
sqrt((Results_static_wifi_distance(wr,1)-
Results_static_bluetooth_distance(br,1))^2+(Results_static_wifi_distance(wr,2)-
Results_static_bluetooth_distance(br,2))^2+(Results_static_wifi_distance(wr,5)-
Results_static_bluetooth_distance(br,5))^2+(Results_static_wifi_distance(wr,6)-
Results_static_bluetooth_distance(br,6))^2);
         % Format: d(w, b) = sqrt((w(s1)-b(s1))^2+(w(s2)-b(s2))^2);
             % d(w, b): is close to 0 if the two rows are very similar and
became greater if the lines are different
             % w: The normalized wifi value
             % b: The normalized bluetooth value
             % s: sensor-1, sensor-2,... sensor-6
     end
 end
     % Top K-values
         data_device_from = 1;
         data_device_to = 10;
     for data_device = 1:10
         % 6 sensors
         current_data =
sortrows(Results_distance_double_signaltodistance_euclideon_6sensors(data_device
_from:data_device_to,:),[3],{'ascend'}); % For every device in
"Results_distance_double_signaltodistance_euclideon", sort the values in
ascending order

TopK_6sensors_distance_double_signaltodistance_euclideon(data_device_from:data_d
evice_to,:) = current_data; % Final values to look through when plotting into
graph
         % 4 sensors
         current_data =
sortrows(Results_distance_double_signaltodistance_euclideon_4sensors(data_device
_from:data_device_to,:),[3],{'ascend'}); % For every device in
"Results_distance_double_signaltodistance_euclideon", sort the values in
ascending order
```

```
TopK_4sensors_distance_double_signaltodistance_euclideon(data_device_from:data_d
evice_to,:) = current_data; % Final values to look through when plotting into
graph
         data_device_from = data_device_from + 10; % sliding windows - next 10
devices
         data_device_to = data_device_to + 10; % sliding windows - next 10
devices
     end
```

## 3 Converting Bluetooth signal to Wi-Fi signal + Euclidaen (equations from logarithmic regression lines) - 4/6 sensors

```
     Converted_signalstrength_bluetooth_to_wifi_double=zeros(10,7);
     for r = 1:10 % loop through all devices
         for s = 1:6 % loop through all sensors
             if groupofdevices == 1 % Known devices - Using linear expressions
(obtained in excel by building graphs from experiment 1 (devices were moved
backwards from 1 meter to 15 meters)
                 if r == 1
                     % Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)-5.097)/1.0727;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*0.4215 - 23.797;
                 elseif r == 2
                     %Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)+44.115)/0.1524;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*1.0895 + 11.264;
                 elseif r == 3
                     %Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)+14.934)/0.71;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*0.6997 - 6.4171;
                 elseif r == 4
                     %Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)+14.85)/0.6798;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*0.6795 - 8.8969;
                 elseif r == 5
                     %Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)+8.9543)/0.9899;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*1.2073 + 22.484;
                 elseif r == 6
```

```
                     %Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)+18.694)/0.6694;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*0.4686 - 16.179;
                 elseif r == 7
                     %Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)+22.306)/0.6028;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*0.8417 - 4.3166;
                 elseif r == 8
                     %Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)+36.307)/0.2334;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*1.0793 + 10.026;
                 elseif r == 9
                     %Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)+15.659)/0.7495;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*0.9074 + 2.9372;
                 elseif r == 10
                     %Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s)+1.969)/1.0533;
                     Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*1.3121 + 21.601;
                 end
             else % Mean linear expression used by the "unknown devices" - See
figure "Wi-Fi vs Bluetooth"
                 Converted_signalstrength_bluetooth_to_wifi_double(r,s+1)=
(Results_static_bluetooth_signalstrength(r,s))*1.0793 - 10.643; % Mean of all 10
devices
             end
             Converted_signalstrength_bluetooth_to_wifi_double(r,1) = r; %
Setting current device in first column
         end
     end
     % Euclideaon distance - Testing each Wi-Fi MAC address against each
Bluetooth MAC address
 length=0;
     for wr=1:10 % loop through the wi-fi data - wr (wifi-row)
         for br=1:10  % loop through the converted bluetooth to wi-fi data - br
(bluetooth-row)
             length = length+1;
             % Calculation based on 6 sensors (1,2,3,4,5,6) - Euclidean distance
between two points (in this case between the original Wi-Fi values and the
converted Wi-Fi values). Saves values in column 3
```

```
Results_converted_signalstrength_double_euclideon_6sensors(length,1) = wr; %
Saves Wi-Fi device number in column 1

Results_converted_signalstrength_double_euclideon_6sensors(length,2) = br; %
Saves Bluetooth device number in column 2

Results_converted_signalstrength_double_euclideon_6sensors(length,3) =
sqrt((Results_static_wifi_signalstrength(wr,1)-
Converted_signalstrength_bluetooth_to_wifi_double(br,1+1))^2+(Results_static_wif
i_signalstrength(wr,2)-
Converted_signalstrength_bluetooth_to_wifi_double(br,2+1))^2+(Results_static_wif
i_signalstrength(wr,3)-
Converted_signalstrength_bluetooth_to_wifi_double(br,3+1))^2+(Results_static_wif
i_signalstrength(wr,4)-
Converted_signalstrength_bluetooth_to_wifi_double(br,4+1))^2+(Results_static_wif
i_signalstrength(wr,5)-
Converted_signalstrength_bluetooth_to_wifi_double(br,5+1))^2+(Results_static_wif
i_signalstrength(wr,6)-
Converted_signalstrength_bluetooth_to_wifi_double(br,6+1))^2);
             % Calculation based on 4 sensors (1,2,5,6) - Euclidean distance
between two points (in this case between the Bluetooth and Wi-Fi values)

Results_converted_signalstrength_double_euclideon_4sensors(length,1) = wr; %
Saves Wi-Fi device number in column 1

Results_converted_signalstrength_double_euclideon_4sensors(length,2) = br; %
Saves Bluetooth device number in column 2

Results_converted_signalstrength_double_euclideon_4sensors(length,3) =
sqrt((Results_static_wifi_signalstrength(wr,1)-
Converted_signalstrength_bluetooth_to_wifi_double(br,1+1))^2+(Results_static_wif
i_signalstrength(wr,2)-
Converted_signalstrength_bluetooth_to_wifi_double(br,2+1))^2+(Results_static_wif
i_signalstrength(wr,5)-
Converted_signalstrength_bluetooth_to_wifi_double(br,5+1))^2+(Results_static_wif
i_signalstrength(wr,6)-
Converted_signalstrength_bluetooth_to_wifi_double(br,6+1))^2);
                 % Format: d(w, b) = sqrt((w(s1)-b(s1))^2+(w(s2)-b(s2))^2);
                 % d(w, b): is close to 0 if the two rows are very similar and
became greater if the lines are different
                 % w: The original singals from wifi
                 % b: The converted bluetooth to wi-fi signal
                 % s: sensor-1, sensor-2,... sensor-6
         end
     end
     % Top K-values
     data_device_from = 1;
     data_device_to = 10;
     for data_device = 1:10
         % 6 sensors
         current_data =
sortrows(Results_converted_signalstrength_double_euclideon_6sensors(data_device_
from:data_device_to,:),[3],{'ascend'}); % For every device in
"Results_signalstrength_double_norm_euclideon", sort the values in ascending
order

TopK_converted_signalstrength_double_euclideon_6sensors(data_device_from:data_de
vice_to,:) = current_data; % Final values to look through when plotting into
graph
         % 4 sensors
         current_data =
sortrows(Results_converted_signalstrength_double_euclideon_4sensors(data_device_
from:data_device_to,:),[3],{'ascend'}); % For every device in
"Results_signalstrength_double_norm_euclideon", sort the values in ascending
order

TopK_converted_signalstrength_double_euclideon_4sensors(data_device_from:data_de
vice_to,:) = current_data; % Final values to look through when plotting into
graph
         data_device_from = data_device_from + 10; % sliding windows - next 10
devices
         data_device_to = data_device_to + 10; % sliding windows - next 10
devices
     end
```

## 4 Triangulation + Euclidean

```
      results_triangulation = zeros(200,5);
      results_triangulation_bluetooth_mean = zeros(10,3);
      results_triangulation_wifi_mean = zeros(10,3);
      Results_triangulation_distancetocoordinates_euclideon = zeros(100,3);
      current_data = Results_static_wifi_distance; % Choosing the wifi data set
      current_row = 1;
     for technologies = 1:2 % loop through both Wi-Fi and Bluetooth data
         current_device = 1; % Starting at device one
         for current_device = 1:size(current_data,1) % Looping through all
devices
           % 1 Triangle A perspective (s1-s2)
             % Check for "true" triangles: Check if distance between s1-s2 >
widt and distance between s5-s1 > length, also that the signal converted to
distance is shorter than 22.5 and that both distances does not exeeds 45
```

```
             if ((current_data(current_device,1)+current_data(current_device,2)
> 11.195) && (current_data(current_device,1)+current_data(current_device,2) <
30.7) && ((current_data(current_device,1) < 15.35) &&
(current_data(current_device,2) < 15.35)))
             % Step 1 - Calculate angle in triangle 1A with the law of cosines:
a^2 = b2^2 + c2^2-2bc*Cos(A) --> Cos(A) = (b2^2+c^2-a^2)/(2*bc)
                    angle_triangle_1A_s1s2 =
((11.195)^2+(current_data(current_device,1))^2-
(current_data(current_device,2))^2)/(2*11.195*(current_data(current_device,1)));
% Law of cosines
             % Step 2 - Calculate the (adjacent) in a right-angled triangle with
the law of cosines: cos(A) = adjacent / hypotenuse, where hyp = a --> cos(A) * a
= adj
                if (angle_triangle_1A_s1s2 < 1 && angle_triangle_1A_s1s2
>cosd(62.5)) % Check angle - Must be between 0 and 45 degrees, filtering out
angles outside of the sector/room
                    adj_triangle_1A_s1s2 =
(angle_triangle_1A_s1s2*current_data(current_device,1)); % adj = angle * c
             % Step 3 - Calculate opposite with pytagoras theorem: c^2 = adj^2 +
opp^2
                    opp_triangle_1A_s1s2 =
sqrt((current_data(current_device,1))^2-(adj_triangle_1A_s1s2)^2); %
             % Storing opp/adj-values in triangulation_results --> (adj,opp)
                    results_triangulation(current_row,1) = technologies; % Put
current device in first column
                    results_triangulation(current_row,2) = current_device;  %
Put current technology in second column
                    results_triangulation(current_row,3) = 1.1; % Put current
triangle in third column
                    results_triangulation(current_row,4) = adj_triangle_1A_s1s2;
% Put current adj value in fourth column as x
                    results_triangulation(current_row,5) = opp_triangle_1A_s1s2;
% Put current opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 1 Triangle B perspective (s2-s1)
             % Check for "true" triangles: Check if distance between s1-s2 >
widt and distance between s5-s1 > length, also that the signal converted to
distance is shorter than 22.5 and that both distances does not exeeds 45
             if ((current_data(current_device,2)+current_data(current_device,1)
> 11.195) && (current_data(current_device,2)+current_data(current_device,1) <
30.7) && ((current_data(current_device,2) < 15.35) &&
(current_data(current_device,1) < 15.35)))
```

```
            % Step 1 - Calculate angle in triangle 1B with the law of cosines:
b^2 = c^2+a^2-2ca*Cos(B) --> Cos(B) = (c^2+a^2-b^2)/2ca
                    angle_triangle_1B_s2s1 =
((current_data(current_device,2))^2+(current_data(current_device,1))^2-
(11.195)^2)/(2*(current_data(current_device,2)*current_data(current_device,1)));
% Law of cosines
            % Step 2 - Calculate the (adjacent) in a right-angled triangle with
the law of cosines: cos(A) = adjacent / hypotenuse, where hyp = a --> cos(A) * a
= adj
                if (angle_triangle_1B_s2s1 < 1 && angle_triangle_1B_s2s1
>cosd(62.5)) % Check angle - Must be between 0 and 1, filtering out angles
outside of the room
                    adj_triangle_1B_s2s1 =
(angle_triangle_1B_s2s1*current_data(current_device,2)); % adj = angle * a
            % Step 3 - Calculate opposite with pytagoras theorem: a^2 = adj^2 +
opp^2
                    opp_triangle_1B_s2s1 =
sqrt((current_data(current_device,2))^2-(adj_triangle_1B_s2s1)^2); %
            % Storing opp/adj-values in triangulation_results --> (adj,opp)
                    results_triangulation(current_row,1) = technologies; % Put
current device in first column
                    results_triangulation(current_row,2) = current_device;  %
Put current technology in second column
                    results_triangulation(current_row,3) = 1.2; % Put current
triangle in third column
                    results_triangulation(current_row,4) = 11.195-
adj_triangle_1B_s2s1; % Put current adj value in fourth column as x
                    results_triangulation(current_row,5) = opp_triangle_1B_s2s1;
% Put current opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 2 Triangle A perspective (s5-s1)
            if ((current_data(current_device,5)+current_data(current_device,1) >
20.8) && (current_data(current_device,5)+current_data(current_device,1) < 30.7)
&& ((current_data(current_device,5) < 15.35) && (current_data(current_device,1)
< 15.35)))
            angle_triangle_2A_s5s1 =
((20.80)^2+(current_data(current_device,5))^2-
(current_data(current_device,1))^2)/(2*20.80*(current_data(current_device,5)));
                if (angle_triangle_2A_s5s1 < 1 && angle_triangle_2A_s5s1
>=cosd(45))
                    adj_triangle_2A_s5s1 =
(angle_triangle_2A_s5s1*current_data(current_device,5)); % y
```

```
                    opp_triangle_2A_s5s1 = 
sqrt((current_data(current_device,5))^2-(adj_triangle_2A_s5s1)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 2.1;
                    results_triangulation(current_row,4) = opp_triangle_2A_s5s1; 
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_2A_s5s1; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 2 Triangle B perspective (s1-s5) - Most inaccurate
            if ((current_data(current_device,1)+current_data(current_device,5) > 
20.8) && (current_data(current_device,1)+current_data(current_device,5) < 30.7) 
&& ((current_data(current_device,1) < 15.35) && (current_data(current_device,5) 
< 15.35)))
            angle_triangle_2B_s1s5 = 
((current_data(current_device,1))^2+(current_data(current_device,5))^2-
(11.195)^2)/(2*(current_data(current_device,1)*current_data(current_device,5)));
                if (angle_triangle_2B_s1s5 < 1 && angle_triangle_2B_s1s5 
>=cosd(45))
                    adj_triangle_2B_s1s5 = 
(angle_triangle_2B_s1s5*current_data(current_device,1)); % y
                    opp_triangle_2B_s1s5 = 
sqrt((current_data(current_device,1))^2-(adj_triangle_2B_s1s5)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 2.2;
                    results_triangulation(current_row,4) = opp_triangle_2B_s1s5; 
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) = adj_triangle_2B_s1s5; 
% Put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 3 Triangle A perspective (s6-s5)
            if ((current_data(current_device,6)+current_data(current_device,5) > 
11.195) && (current_data(current_device,6)+current_data(current_device,5) < 
30.7) && ((current_data(current_device,6) < 15.35) && 
(current_data(current_device,5) < 15.35)))
```

```
            angle_triangle_3A_s6s5 =
((11.195)^2+(current_data(current_device,6))^2-
(current_data(current_device,5))^2)/(2*11.195*(current_data(current_device,6)));
                if (angle_triangle_3A_s6s5 < 1 && angle_triangle_3A_s6s5
>cosd(65))
                    adj_triangle_3A_s6s5 =
(angle_triangle_3A_s6s5*current_data(current_device,6)); % x
                    opp_triangle_3A_s6s5 =
sqrt((current_data(current_device,6))^2-(adj_triangle_3A_s6s5)^2); % y
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 3.1;
                    results_triangulation(current_row,4) = 11.195-
adj_triangle_3A_s6s5; % Need to subtract adj from width to obtain same x/y-axis,
then put adj value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
opp_triangle_3A_s6s5; % Need to subtract opp from length to obtain same x/y-
axis, then put opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
       % 3 Triangle B perspective (s5-s6)
            if ((current_data(current_device,5)+current_data(current_device,6) >
11.195) && (current_data(current_device,5)+current_data(current_device,6) <
30.7) && ((current_data(current_device,5) < 15.35) &&
(current_data(current_device,6) < 15.35)))
            angle_triangle_3B_s5s6 =
((current_data(current_device,5))^2+(current_data(current_device,6))^2-
(11.195)^2)/(2*(current_data(current_device,5)*current_data(current_device,6)));
                if (angle_triangle_3B_s5s6 < 1 && angle_triangle_3B_s5s6
>cosd(65))
                    adj_triangle_3B_s5s6 =
(angle_triangle_3B_s5s6*current_data(current_device,5)); % x
                    opp_triangle_3B_s5s6 =
sqrt((current_data(current_device,5))^2-(adj_triangle_3B_s5s6)^2); % y
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 3.2;
                    results_triangulation(current_row,4) = adj_triangle_3B_s5s6;
% Put adj value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
opp_triangle_3B_s5s6; % Need to subtract opp from length to obtain same x/y-
axis, then put opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 4 Triangle A perspective (s2-s6) Not so accurate
            if ((current_data(current_device,2)+current_data(current_device,6) >
20.8) && (current_data(current_device,2)+current_data(current_device,6) < 30.7)
&& ((current_data(current_device,2) < 15.35) && (current_data(current_device,6)
< 15.35)))
            angle_triangle_4A_s2s6 =
((20.80)^2+(current_data(current_device,2))^2-
(current_data(current_device,6))^2)/(2*20.80*(current_data(current_device,2)));
                if (angle_triangle_4A_s2s6 < 1 && angle_triangle_4A_s2s6
>=cosd(45))
                    adj_triangle_4A_s2s6 =
(angle_triangle_4A_s2s6*current_data(current_device,2)); % y
                    opp_triangle_4A_s2s6 =
sqrt((current_data(current_device,2))^2-(adj_triangle_4A_s2s6)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 4.1;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_4A_s2s6; % Need to subtract opp from width to obtain same x/y-axis,
then put adj value in fourth column as x
                    results_triangulation(current_row,5) = adj_triangle_4A_s2s6;
% Put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
          % 4 Triangle B perspective (s6-s2) Not so accurate
            if ((current_data(current_device,6)+current_data(current_device,2) >
20.8) && (current_data(current_device,6)+current_data(current_device,2) < 30.7)
&& ((current_data(current_device,6) < 15.35) && (current_data(current_device,2)
< 15.35)))
            angle_triangle_4B_s6s2 =
((current_data(current_device,6))^2+(current_data(current_device,2))^2-
(11.195)^2)/(2*(current_data(current_device,6)*current_data(current_device,2)));
                if (angle_triangle_4B_s6s2 < 1 && angle_triangle_4B_s6s2
>=cosd(45))
                    adj_triangle_4B_s6s2 =
(angle_triangle_4B_s6s2*current_data(current_device,6)); % y
                    opp_triangle_4B_s6s2 =
sqrt((current_data(current_device,6))^2-(adj_triangle_4B_s6s2)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 4.2;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_4B_s6s2; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_4B_s6s2; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 5 Triangle A perspective (s3-s1) (rotated)
            if ((current_data(current_device,3)+current_data(current_device,1) >
10.40) && (current_data(current_device,3)+current_data(current_device,1) < 15.7)
&& ((current_data(current_device,3) < 11.195) && (current_data(current_device,1)
< 11.195)))
            angle_triangle_5A_s3s1 =
((10.4)^2+(current_data(current_device,3))^2-
(current_data(current_device,1))^2)/(2*10.4*(current_data(current_device,3)));
                if (angle_triangle_5A_s3s1 < 1 && angle_triangle_5A_s3s1 >0)
                    adj_triangle_5A_s3s1 =
(angle_triangle_5A_s3s1*current_data(current_device,3)); % y
                    opp_triangle_5A_s3s1 =
sqrt((current_data(current_device,3))^2-(adj_triangle_5A_s3s1)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 5.1;
                    results_triangulation(current_row,4) = opp_triangle_5A_s3s1;
% Put opp value in fourth column as x (triangle is rotated)
                    results_triangulation(current_row,5) = 10.40-
adj_triangle_5A_s3s1; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y (triangle is rotated)
                    current_row = current_row +1;
                else
                end
            end
         % 5 Triangle B perspective (s1-s3)
            if ((current_data(current_device,1)+current_data(current_device,3) >
10.40) && (current_data(current_device,1)+current_data(current_device,3) < 15.7)
&& ((current_data(current_device,1) < 11.195) && (current_data(current_device,3)
< 11.195)))
            angle_triangle_5B_s1s3 =
((current_data(current_device,1))^2+(current_data(current_device,3))^2-
(11.195)^2)/(2*(current_data(current_device,1)*current_data(current_device,3)));
                if (angle_triangle_5B_s1s3 < 1 && angle_triangle_5B_s1s3 >0)
                    adj_triangle_5B_s1s3 =
(angle_triangle_5B_s1s3*current_data(current_device,1)); % y
```

```
                    opp_triangle_5B_s1s3 = 
sqrt((current_data(current_device,1))^2-(adj_triangle_5B_s1s3)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 5.2;
                    results_triangulation(current_row,4) = opp_triangle_5B_s1s3; 
% Put opp value in fourth column as x (triangle is rotated)
                    results_triangulation(current_row,5) = adj_triangle_5B_s1s3; 
% Put adj value in fifth column as y (triangle is rotated)
                    current_row = current_row +1;
                else
                end
            end
          % 6 Triangle A perspective (s5-s3)
            if ((current_data(current_device,5)+current_data(current_device,3) > 
10.40) && (current_data(current_device,5)+current_data(current_device,3) < 15.7) 
&& ((current_data(current_device,5) < 11.195) && (current_data(current_device,3) 
< 11.195)))
            angle_triangle_6A_s5s3 = 
((10.4)^2+(current_data(current_device,5))^2-
(current_data(current_device,3))^2)/(2*10.4*(current_data(current_device,5)));
                if (angle_triangle_6A_s5s3 < 1 && angle_triangle_6A_s5s3 >0)
                    adj_triangle_6A_s5s3 = 
(angle_triangle_6A_s5s3*current_data(current_device,5)); % y
                    opp_triangle_6A_s5s3 = 
sqrt((current_data(current_device,5))^2-(adj_triangle_6A_s5s3)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 6.1;
                    results_triangulation(current_row,4) = opp_triangle_6A_s5s3; 
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_6A_s5s3; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 6 Triangle B perspective (s3-s5)
            if ((current_data(current_device,3)+current_data(current_device,5) > 
10.40) && (current_data(current_device,3)+current_data(current_device,5) < 15.7) 
&& ((current_data(current_device,3) < 11.195) && (current_data(current_device,5) 
< 11.195)))
            angle_triangle_6B_s3s5 = 
((current_data(current_device,3))^2+(current_data(current_device,5))^2-
(11.195)^2)/(2*(current_data(current_device,3)*current_data(current_device,5)));
```

```
                if (angle_triangle_6B_s3s5 < 1 && angle_triangle_6B_s3s5 >0)
                    adj_triangle_6B_s3s5 =
(angle_triangle_6B_s3s5*current_data(current_device,3)); % y
                    opp_triangle_6B_s3s5 =
sqrt((current_data(current_device,3))^2-(adj_triangle_6B_s3s5)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 6.2;
                    results_triangulation(current_row,4) = opp_triangle_6B_s3s5;
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) =
10.40+adj_triangle_6B_s3s5; % Need to add length to adj to obtain same x/y-axis,
then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
           % 7 Triangle A perspective (s4-s6)
            if ((current_data(current_device,4)+current_data(current_device,6) >
10.40) && (current_data(current_device,4)+current_data(current_device,6) < 15.7)
&& ((current_data(current_device,4) < 11.195) && (current_data(current_device,6)
< 11.195)))
            angle_triangle_7A_s4s6 =
((10.4)^2+(current_data(current_device,4))^2-
(current_data(current_device,6))^2)/(2*10.4*(current_data(current_device,4)));
                if (angle_triangle_7A_s4s6 < 1 && angle_triangle_7A_s4s6 >0)
                    adj_triangle_7A_s4s6 =
(angle_triangle_7A_s4s6*current_data(current_device,4)); % y
                    opp_triangle_7A_s4s6 =
sqrt((current_data(current_device,4))^2-(adj_triangle_7A_s4s6)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 7.1;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_7A_s4s6; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) =
10.40+adj_triangle_7A_s4s6; % Need to add length to adj to obtain same x/y-axis,
then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
            % 7 Triangle B perspective (s6-s4)
            if ((current_data(current_device,6)+current_data(current_device,4) >
10.40) && (current_data(current_device,6)+current_data(current_device,4) < 15.7)
&& ((current_data(current_device,6) < 11.195) && (current_data(current_device,4)
< 11.195)))
            angle_triangle_7B_s6s4 =
((current_data(current_device,6))^2+(current_data(current_device,4))^2-
(11.195)^2)/(2*(current_data(current_device,6)*current_data(current_device,4)));
                if (angle_triangle_7B_s6s4 < 1 && angle_triangle_7B_s6s4 >0)
                    adj_triangle_7B_s6s4 =
(angle_triangle_7B_s6s4*current_data(current_device,6)); % y
                    opp_triangle_7B_s6s4 =
sqrt((current_data(current_device,6))^2-(adj_triangle_7B_s6s4)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 7.2;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_7B_s6s4; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_7B_s6s4; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
          % 8 Triangle A perspective (s2-s4)
            if ((current_data(current_device,2)+current_data(current_device,4) >
10.40) && (current_data(current_device,2)+current_data(current_device,4) < 15.7)
&& ((current_data(current_device,2) < 11.195) && (current_data(current_device,4)
< 11.195)))
            angle_triangle_8A_s2s4 =
((10.4)^2+(current_data(current_device,2))^2-
(current_data(current_device,4))^2)/(2*10.4*(current_data(current_device,2)));
                if (angle_triangle_8A_s2s4 < 1 && angle_triangle_8A_s2s4 >0)
                    adj_triangle_8A_s2s4 =
(angle_triangle_8A_s2s4*current_data(current_device,2)); % Need to calculate y
first (the triangle is rotated).
                    opp_triangle_8A_s2s4 =
sqrt((current_data(current_device,2))^2-(adj_triangle_8A_s2s4)^2);
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 8.1;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_8A_s2s4; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = adj_triangle_8A_s2s4;
% Put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 8 Triangle B perspective (s4-s2)
            if ((current_data(current_device,4)+current_data(current_device,2) >
10.40) && (current_data(current_device,4)+current_data(current_device,2) < 15.7)
&& ((current_data(current_device,4) < 11.195) && (current_data(current_device,2)
< 11.195)))
            angle_triangle_8B_s4s2 =
((current_data(current_device,4))^2+(current_data(current_device,2))^2-
(11.195)^2)/(2*(current_data(current_device,4)*current_data(current_device,2)));
                if (angle_triangle_8B_s4s2 < 1 && angle_triangle_8B_s4s2 >0)
                    adj_triangle_8B_s4s2 =
(angle_triangle_8B_s4s2*current_data(current_device,4)); % y
                    opp_triangle_8B_s4s2 =
sqrt((current_data(current_device,4))^2-(adj_triangle_8B_s4s2)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 8.2;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_8B_s4s2; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = 10.40-
adj_triangle_8B_s4s2; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
           % Remove negative values in results_triangulation and results outside
of room
           results_triangulation =
results_triangulation(results_triangulation(:,4)>0,:);
           results_triangulation =
results_triangulation(results_triangulation(:,5)>0,:);
           results_triangulation =
results_triangulation(results_triangulation(:,4)<11.195,:);
           results_triangulation =
results_triangulation(results_triangulation(:,5)<20.80,:);
           current_row = size(find(results_triangulation(:,1)>0),1)+1; % Find
total rows after removing negative values
       % Pre-work Euclideaon distance - If results greater than 0, calculate
mean values (coordinate) among all triangles. Do it for both Wifi and Blueototh
data
           currentresults =
results_triangulation((results_triangulation(:,1)==technologies) &
(results_triangulation(:,2)==current_device),:);  % Save results that only
contain current device and current technology
                 %if size(currentresults)>=1 % Test to see if data is zero
                     if technologies == 2 % Test to see if wifi or bluetooth
                       xy_results_triangulation_bluetooth_currentlocation =
currentresults(:,3:5); % Use x and y value in column 4 and 5
                       % Bluetooth - If there exists multiple results for the
same device, calculate mean value of these coordinates
                       count_current_values =
size(xy_results_triangulation_bluetooth_currentlocation(:,1));
                         if count_current_values(1) >= 1 % if there exists
multiple results, then statement is true

results_triangulation_bluetooth_mean(current_device,1) = currentresults(1,2); %
put device number in first column

results_triangulation_bluetooth_mean(current_device,2) =
mean(xy_results_triangulation_bluetooth_currentlocation(:,2)); % mean of x
values

results_triangulation_bluetooth_mean(current_device,3) =
mean(xy_results_triangulation_bluetooth_currentlocation(:,3)); % mean of y
values
                             results_triangulation_bluetooth_mean =
results_triangulation_bluetooth_mean(results_triangulation_bluetooth_mean(:,1)>0
,:); % remove null values
                         else % Put null values in the table (will also be
compared)

results_triangulation_bluetooth_mean(current_device,1)=current_device;

results_triangulation_bluetooth_mean(current_device,2:3)=nan;
                         end
                     else
                       xy_results_triangulation_wifi_currentlocation =
currentresults(:,3:5);
                       % WiFi - If there exists multiple results for the same
second, calculate mean value of these coordinates
                       count_current_values =
size(xy_results_triangulation_wifi_currentlocation(:,1));
                         if count_current_values(1) >= 1 % if there exists
multiple results, then statement is true
                             results_triangulation_wifi_mean(current_device,1) =
currentresults(1,2); % put device number in first column
                             results_triangulation_wifi_mean(current_device,2) =
mean(xy_results_triangulation_wifi_currentlocation(:,2)); % mean of x values
```

```
                             results_triangulation_wifi_mean(current_device,3) =
mean(xy_results_triangulation_wifi_currentlocation(:,3)); % mean of y values
                             results_triangulation_wifi_mean =
results_triangulation_wifi_mean(results_triangulation_wifi_mean(:,1)>0,:); %
remove null values
                         else % Put null values in the table (will also be
compared)

results_triangulation_wifi_mean(current_device,1)=current_device;

results_triangulation_wifi_mean(current_device,2:3)=nan;
                         end
                     end
         end
         current_data = Results_static_bluetooth_distance; % Changing technology
and data set
     end

        % Euclideaon distance - Testing best coordinate obtained above (in
relation to known locations) between Wi-fi and Bluetooth
     length=0;
     for wr=1:10 % loop through the calculated coordinates from wifi data - wr
(wifi-row)
         for br=1:10  % loop through the calculated coordinates from bluetooth
data - br (bluetooth-row)
             length = length+1; % Increment length by one on each loop
             Results_triangulation_distancetocoordinates_euclideon(length,1) =
wr; % Saves Wi-Fi device number in column 1
             Results_triangulation_distancetocoordinates_euclideon(length,2) =
br; % Saves Bluetooth device number in column 2
             % Calculates the Euclidean distance between two points (in this
case between the nearest Bluetooth and Wi-Fi coordinates in relation to known
locations)
             Results_triangulation_distancetocoordinates_euclideon(length,3) =
sqrt((results_triangulation_wifi_mean(wr,2)-
results_triangulation_bluetooth_mean(br,2))^2+(results_triangulation_wifi_mean(w
r,3)-results_triangulation_bluetooth_mean(br,3))^2);
             % Format: d(w, b) = sqrt((w(s1)-b(s1))^2+(w(s2)-b(s2))^2);
         end
     end

     % Top K-values 6 sensors (mean value of all triangles)
         data_device_from = 1;
         data_device_to = 10;
     for data_device = 1:10
```

```
         current_data =
sortrows(Results_triangulation_distancetocoordinates_euclideon(data_device_from:
data_device_to,:),[3],{'ascend'}); % For every device in
"Results_triangulation_distancetocoordinates_euclideon", sort the values in
ascending order

TopK_triangulation_distancetocoordinates_euclideon(data_device_from:data_device_
to,:) = current_data; % Final values to look through when plotting into graph
         data_device_from = data_device_from + 10;
         data_device_to = data_device_to + 10;
     end
```

## 5 Trilateration + Euclidean

```
     xy_results1 = zeros(10,4);
     xy_results2 = zeros(10,4);
     xy_results3 = zeros(10,4);
     xy_results4 = zeros(10,4);

     xy_results_trilateration_wifi_total = zeros(200,4);
     xy_results_trilateration_bluetooth_total = zeros(200,4);
     xy_results_trilateration_wifi_mean = zeros(10,3);
     xy_results_trilateration_bluetooth_mean = zeros(10,3);
     Data=Results_static_wifi_distance;

     for technologies = 1:2 % loop through both Wi-Fi and Bluetooth data
           length_of_data = 1;
         for length_of_data = 1:size(Data,1)

         % Stores values from sensor 1,2,5,6 based on current row into r1..r6
          r1 = (Data(length_of_data,1));
          r2 = (Data(length_of_data,2));
          r5 = (Data(length_of_data,5));
          r6 = (Data(length_of_data,6));

     % See how step 1-4 were performed in method 2 Trilateration
     % Step 5 - Let Matlab understand the three equations

             A = 2*(-xy(1,1)+xy(1,2));
             B = 2*(-xy(2,1)+xy(2,2));
             C = (r1)^2-(r2)^2-(xy(1,1))^2+(xy(1,2))^2-(xy(2,1))^2+(xy(2,2))^2;

             D = 2*(-xy(1,2)+xy(1,3));
             E = 2*(-xy(2,2)+xy(2,3));
             F = (r2)^2-(r5)^2-(xy(1,2))^2+(xy(1,3))^2-(xy(2,2))^2+(xy(2,3))^2;
```

```
            G = 2*(-xy(1,3)+xy(1,4));
            H = 2*(-xy(2,3)+xy(2,4));
            I = (r5)^2-(r6)^2-(xy(1,3))^2+(xy(1,4))^2-(xy(2,3))^2+(xy(2,4))^2;

            J = 2*(-xy(1,4)+xy(1,1));
            K = 2*(-xy(2,4)+xy(2,1));
            L = (r6)^2-(r1)^2-(xy(1,4))^2+(xy(1,1))^2-(xy(2,4))^2+(xy(2,1))^2;

            % 1 Equation 1 (Sensor 1,2,5)
            x_1 = abs((C*E-B*F)/(A*E-B*D));
            y_1 = abs((C*D-A*F)/(B*D-A*E));

            % 2 Equation 2 (Sensor 2,5,6)
            x_2 = abs((F*H-E*I)/(D*H-E*G));
            y_2 = abs((F*G-D*I)/(E*G-D*H));

            % 3 Equation 3 (Sensor 5,6,1)
            x_3 = abs((I*K-H*L)/(G*K-H*J));
            y_3 = abs((I*J-G*L)/(H*J-G*K));

            % 4 Equation 4 (Sensor 6-1-2)
            x_4 = abs((L*A-K*C)/(J*B-K*A));
            y_4 = abs((L*A-J*C)/(K*A-J*B));

            % Save values of x and y into tables of results based on different
equations, remove values outside of room or null values
            if x_1 > 11.195 || y_1 > 21.5 || isnan(x_1) || isnan(y_1) %
Original distance is 20.8. 21.5 meters is to include data from location 9 in
static experiment 3-1
            else
                xy_results1(length_of_data,1) = x_1; % Stores x values in table
xy_results1 from equation 1
                xy_results1(length_of_data,2) = y_1; % Stores y values in table
xy_results1 from equation 1
                xy_results1(length_of_data,3) = 1; % Store equation number in
column 3
                xy_results1(length_of_data,4) = length_of_data; % Store device
number in column 4
            end
            if x_2 > 11.195 || y_2 > 21.5 || isnan(x_2) || isnan(y_2)
            else
                xy_results2(length_of_data,1) = x_2; % Stores x values in table
xy_results2 from equation 2
                xy_results2(length_of_data,2) = y_2; % Stores y values in table
xy_results2 from equation 2
```

```
                 xy_results2(length_of_data,3) = 2; % Store equation number in
column 3
                 xy_results2(length_of_data,4) = length_of_data; % Store device
number in column 4
             end
             if x_3 > 11.195 || y_3 > 21.5 || isnan(x_3) || isnan(y_3)
             else
                 xy_results3(length_of_data,1) = x_3; % Stores x values in table
xy_results3 from equation 3
                 xy_results3(length_of_data,2) = y_3; % Stores y values in table
xy_results3 from equation 3
                 xy_results3(length_of_data,3) = 3; % Store equation number in
column 3
                 xy_results3(length_of_data,4) = length_of_data; % Store device
number in column 4
             end
             if x_4 > 11.195 || y_4 > 21.5 || isnan(x_4) || isnan(y_4)
             else
                 xy_results4(length_of_data,1) = x_4; % Stores x values in table
xy_results4 from equation 4
                 xy_results4(length_of_data,2) = y_4; % Stores y values in table
xy_results4 from equation 4
                 xy_results4(length_of_data,3) = 4; % Store equation number in
column 3
                 xy_results4(length_of_data,4) = length_of_data; % Store device
number in column 4
             end

                 % Finding mean value of xy_results1-4 related to static
location
                 if technologies == 2
                   xy_results_trilateration_bluetooth_total =
[xy_results1(length_of_data,:); xy_results2(length_of_data,:);
xy_results3(length_of_data,:); xy_results4(length_of_data,:)];
                   xy_results_trilateration_bluetooth_total =
xy_results_trilateration_bluetooth_total(xy_results_trilateration_bluetooth_tota
l(:,1)>0,:); % remove null values
                 % Bluetooth - If there exists multiple results for the same
second, calculate mean value of these coordinates
                   count_current_values =
size(xy_results_trilateration_bluetooth_total(:,1));
                     if count_current_values(1) >= 1 % if there exists multiple
results, then statement is true

xy_results_trilateration_bluetooth_mean(length_of_data,1) =
xy_results_trilateration_bluetooth_total(1,4); % put device number in first
column

xy_results_trilateration_bluetooth_mean(length_of_data,2) =
mean(xy_results_trilateration_bluetooth_total(:,1)); % mean of x values

xy_results_trilateration_bluetooth_mean(length_of_data,3) =
mean(xy_results_trilateration_bluetooth_total(:,2)); % mean of y values
                         xy_results_trilateration_bluetooth_mean =
xy_results_trilateration_bluetooth_mean(xy_results_trilateration_bluetooth_mean(
:,1)>0,:); % remove null values
                     else

xy_results_trilateration_bluetooth_mean(length_of_data,1) = length_of_data; %
put device number in first column

xy_results_trilateration_bluetooth_mean(length_of_data,2:3) = nan;
                         continue; % Passes control to the next iteration of
lengthof_data is zero
                     end
                 else
                   xy_results_trilateration_wifi_total =
[xy_results1(length_of_data,:); xy_results2(length_of_data,:);
xy_results3(length_of_data,:); xy_results4(length_of_data,:)];
                   xy_results_trilateration_wifi_total =
xy_results_trilateration_wifi_total(xy_results_trilateration_wifi_total(:,1)>0,:
); % remove null values
                   % WiFi - If there exists multiple results for the same
second, calculate mean value of these coordinates
                   count_current_values =
size(xy_results_trilateration_wifi_total(:,1));
                     if count_current_values(1) >= 1 % if there exists multiple
results, then statement is true
                         xy_results_trilateration_wifi_mean(length_of_data,1) =
xy_results_trilateration_wifi_total(1,4); % put device number in first column
                         xy_results_trilateration_wifi_mean(length_of_data,2) =
mean(xy_results_trilateration_wifi_total(:,1)); % mean of x values
                         xy_results_trilateration_wifi_mean(length_of_data,3) =
mean(xy_results_trilateration_wifi_total(:,2)); % mean of y values
                         xy_results_trilateration_wifi_mean =
xy_results_trilateration_wifi_mean(xy_results_trilateration_wifi_mean(:,1)>0,:);
% remove null values
                     else
                         xy_results_trilateration_wifi_mean(length_of_data,1) =
length_of_data; % put device number in first column
```

```
                         xy_results_trilateration_wifi_mean(length_of_data,2:3)
= nan;
                         continue; % Passes control to the next iteration of
lengthof_data is zero
                     end
                 end
         end
             length_of_data = length_of_data + 1;

                 % Save results from each equation into seperate variables
                 if technologies == 2
                     xy_results_trilateration_bluetooth_1 =
xy_results1(xy_results1(:,1)>0,:); % Save only valid coordiantes (0,0 is
invalid)
                     xy_results_trilateration_bluetooth_2 =
xy_results2(xy_results2(:,1)>0,:);
                     xy_results_trilateration_bluetooth_3 =
xy_results3(xy_results3(:,1)>0,:);
                     xy_results_trilateration_bluetooth_4 =
xy_results4(xy_results4(:,1)>0,:);
                 else
                     Data=Results_static_bluetooth_distance;
                     xy_results_trilateration_wifi_1 =
xy_results1(xy_results1(:,1)>0,:);
                     xy_results_trilateration_wifi_2 =
xy_results2(xy_results2(:,1)>0,:);
                     xy_results_trilateration_wifi_3 =
xy_results3(xy_results3(:,1)>0,:);
                     xy_results_trilateration_wifi_4 =
xy_results4(xy_results4(:,1)>0,:);
                     xy_results1 = zeros(10,4);   % resetting before bluetooth
technology
                     xy_results2 = zeros(10,4);
                     xy_results3 = zeros(10,4);
                     xy_results4 = zeros(10,4);
                 end
     end
     % Concatenating all bluetooth and wifi results into two tables
     xy_results_trilateration_wifi_total =
cat(1,xy_results_trilateration_wifi_1,xy_results_trilateration_wifi_2,xy_results
_trilateration_wifi_3,xy_results_trilateration_wifi_4);
     xy_results_trilateration_bluetooth_total =
cat(1,xy_results_trilateration_bluetooth_1,xy_results_trilateration_bluetooth_2,
xy_results_trilateration_bluetooth_3,xy_results_trilateration_bluetooth_4);
```

```
    % Euclideaon distance 4 sensors (mean value)- Testing each coordinate from
Wi-fi data with each coordinate from Bluetooth
     length=0;
     for wr=1:10 % loop through the calculated coordinates from wifi data - wr
(wifi-row)
         for br=1:10  % loop through the calculated coordinates from bluetooth
data - br (bluetooth-row)
             length = length+1; % Increment length by one on each loop
             Results_multilateration_distancetocoordinates_euclideon(length,1) =
wr; % Saves Wi-Fi device number in column 1
             Results_multilateration_distancetocoordinates_euclideon(length,2) =
br; % Saves Bluetooth device number in column 2
                 % Calculates the Euclidean distance between two points (in this
case between the Bluetooth and Wi-Fi values)
             Results_multilateration_distancetocoordinates_euclideon(length,3) =
sqrt((xy_results_trilateration_wifi_mean(wr,2)-
xy_results_trilateration_bluetooth_mean(br,2))^2+(xy_results_trilateration_wifi_
mean(wr,3)-xy_results_trilateration_bluetooth_mean(br,3))^2);

             %Results_triangulation_distancetocoordinates_euclideon(length,3) =
sqrt((results_triangulation_wifi_mean(wr,2)-
results_triangulation_bluetooth_mean(br,2))^2+(results_triangulation_wifi_mean(w
r,3)-results_triangulation_bluetooth_mean(br,3))^2);

             % Format: d(w, b) = sqrt((w(s1)-b(s1))^2+(w(s2)-b(s2))^2);
             % d(w, b): is close to 0 if the two rows are very similar and
became greater if the lines are different
             % w: The coordinates of wifi date
             % b: The coordinates of bluetooth data
             % s: wifi(x1), wifi(y1), bluetooth(x1), bluetooth(x2).
              % Better solution
                     % coordinates = [0,0;6.7028,11.4674];
                     % d = pdist(coordinates,'euclidean');
         end
     end

     % Top K-values
         data_device_from = 1;
         data_device_to = 10;
     for data_device = 1:10
         % Results with all equations - Mean
         current_data =
sortrows(Results_multilateration_distancetocoordinates_euclideon(data_device_fro
m:data_device_to,:),[3],{'ascend'}); % For every device in
"Results_multilateration_distancetocoordinates_euclideon", sort the values in
ascending order
```

```
TopK_multilateration_distancetocoordinates_euclideon(data_device_from:data_devic
e_to,:) = current_data; % Final values to look through when plotting into graph
         data_device_from = data_device_from + 10;
         data_device_to = data_device_to + 10;
     end

     elseif method == 2
```

## Triangulation

```
      results_triangulation = zeros(200,6);
      results_triangulation_bluetooth_mean = zeros(10,3);
      results_triangulation_wifi_mean = zeros(10,3);
      air_distance = zeros(10,3);
      current_data = Results_static_wifi_distance; % Choosing the wifi data set
      current_row = 1;
     for technologies = 1:2 % loop through both Wi-Fi and Bluetooth data
         current_device = 1; % Starting at device one
         for current_device = 1:size(current_data,1) % Looping through all
devices
           % 1 Triangle A perspective (s1-s2)
             % Check for "true" triangles: Check if distance between s1-s2 >
widt and distance between s5-s1 > length, also that the signal converted to
distance is shorter than 22.5 and that both distances does not exeeds 45
             if ((current_data(current_device,1)+current_data(current_device,2)
> 11.195) && (current_data(current_device,1)+current_data(current_device,2) <
30.7) && ((current_data(current_device,1) < 15.35) &&
(current_data(current_device,2) < 15.35)))
             % Step 1 - Calculate angle in triangle 1A with the law of cosines:
a^2 = b2^2 + c2^2-2bc*Cos(A) --> Cos(A) = (b2^2+c^2-a^2)/(2*bc)
                    angle_triangle_1A_s1s2 =
((11.195)^2+(current_data(current_device,1))^2-
(current_data(current_device,2))^2)/(2*11.195*(current_data(current_device,1)));
% Law of cosines
             % Step 2 - Calculate the (adjacent) in a right-angled triangle with
the law of cosines: cos(A) = adjacent / hypotenuse, where hyp = a --> cos(A) * a
= adj
                if (angle_triangle_1A_s1s2 < 1 && angle_triangle_1A_s1s2
>cosd(62.5)) % Check angle - Must be between 0 and 45 degrees, filtering out
angles outside of the sector/room
                    adj_triangle_1A_s1s2 =
(angle_triangle_1A_s1s2*current_data(current_device,1)); % adj = angle * c
             % Step 3 - Calculate opposite with pytagoras theorem: c^2 = adj^2 +
opp^2
```

```
                    opp_triangle_1A_s1s2 =
sqrt((current_data(current_device,1))^2-(adj_triangle_1A_s1s2)^2); %
             % Storing opp/adj-values in triangulation_results --> (adj,opp)
                    results_triangulation(current_row,1) = technologies; % Put
current device in first column
                    results_triangulation(current_row,2) = current_device;  %
Put current technology in second column
                    results_triangulation(current_row,3) = 1.1; % Put current
triangle in third column
                    results_triangulation(current_row,4) = adj_triangle_1A_s1s2;
% Put current adj value in fourth column as x
                    results_triangulation(current_row,5) = opp_triangle_1A_s1s2;
% Put current opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 1 Triangle B perspective (s2-s1)
             % Check for "true" triangles: Check if distance between s1-s2 >
widt and distance between s5-s1 > length, also that the signal converted to
distance is shorter than 22.5 and that both distances does not exeeds 45
             if ((current_data(current_device,2)+current_data(current_device,1)
> 11.195) && (current_data(current_device,2)+current_data(current_device,1) <
30.7) && ((current_data(current_device,2) < 15.35) &&
(current_data(current_device,1) < 15.35)))
             % Step 1 - Calculate angle in triangle 1B with the law of cosines:
b^2 = c^2+a^2-2ca*Cos(B) --> Cos(B) = (c^2+a^2-b^2)/2ca
                    angle_triangle_1B_s2s1 =
((current_data(current_device,2))^2+(current_data(current_device,1))^2-
(11.195)^2)/(2*(current_data(current_device,2)*current_data(current_device,1)));
% Law of cosines
             % Step 2 - Calculate the (adjacent) in a right-angled triangle with
the law of cosines: cos(A) = adjacent / hypotenuse, where hyp = a --> cos(A) * a
= adj
                if (angle_triangle_1B_s2s1 < 1 && angle_triangle_1B_s2s1
>cosd(62.5)) % Check angle - Must be between 0 and 1, filtering out angles
outside of the room
                    adj_triangle_1B_s2s1 =
(angle_triangle_1B_s2s1*current_data(current_device,2)); % adj = angle * a
             % Step 3 - Calculate opposite with pytagoras theorem: a^2 = adj^2 +
opp^2
                    opp_triangle_1B_s2s1 =
sqrt((current_data(current_device,2))^2-(adj_triangle_1B_s2s1)^2); %
             % Storing opp/adj-values in triangulation_results --> (adj,opp)
                    results_triangulation(current_row,1) = technologies; % Put
current device in first column
```

```
                    results_triangulation(current_row,2) = current_device;  %
Put current technology in second column
                    results_triangulation(current_row,3) = 1.2; % Put current
triangle in third column
                    results_triangulation(current_row,4) = 11.195-
adj_triangle_1B_s2s1; % Put current adj value in fourth column as x
                    results_triangulation(current_row,5) = opp_triangle_1B_s2s1;
% Put current opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 2 Triangle A perspective (s5-s1)
            if ((current_data(current_device,5)+current_data(current_device,1) >
20.8) && (current_data(current_device,5)+current_data(current_device,1) < 30.7)
&& ((current_data(current_device,5) < 15.35) && (current_data(current_device,1)
< 15.35)))
            angle_triangle_2A_s5s1 =
((20.80)^2+(current_data(current_device,5))^2-
(current_data(current_device,1))^2)/(2*20.80*(current_data(current_device,5)));
                if (angle_triangle_2A_s5s1 < 1 && angle_triangle_2A_s5s1
>=cosd(45))
                    adj_triangle_2A_s5s1 =
(angle_triangle_2A_s5s1*current_data(current_device,5)); % y
                    opp_triangle_2A_s5s1 =
sqrt((current_data(current_device,5))^2-(adj_triangle_2A_s5s1)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 2.1;
                    results_triangulation(current_row,4) = opp_triangle_2A_s5s1;
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_2A_s5s1; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 2 Triangle B perspective (s1-s5) - Most inaccurate
            if ((current_data(current_device,1)+current_data(current_device,5) >
20.8) && (current_data(current_device,1)+current_data(current_device,5) < 30.7)
&& ((current_data(current_device,1) < 15.35) && (current_data(current_device,5)
< 15.35)))
            angle_triangle_2B_s1s5 =
((current_data(current_device,1))^2+(current_data(current_device,5))^2-
(11.195)^2)/(2*(current_data(current_device,1)*current_data(current_device,5)));
```

```
                if (angle_triangle_2B_s1s5 < 1 && angle_triangle_2B_s1s5
>=cosd(45))
                    adj_triangle_2B_s1s5 =
(angle_triangle_2B_s1s5*current_data(current_device,1)); % y
                    opp_triangle_2B_s1s5 =
sqrt((current_data(current_device,1))^2-(adj_triangle_2B_s1s5)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 2.2;
                    results_triangulation(current_row,4) = opp_triangle_2B_s1s5;
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) = adj_triangle_2B_s1s5;
% Put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 3 Triangle A perspective (s6-s5)
            if ((current_data(current_device,6)+current_data(current_device,5) >
11.195) && (current_data(current_device,6)+current_data(current_device,5) <
30.7) && ((current_data(current_device,6) < 15.35) &&
(current_data(current_device,5) < 15.35)))
            angle_triangle_3A_s6s5 =
((11.195)^2+(current_data(current_device,6))^2-
(current_data(current_device,5))^2)/(2*11.195*(current_data(current_device,6)));
                if (angle_triangle_3A_s6s5 < 1 && angle_triangle_3A_s6s5
>cosd(65))
                    adj_triangle_3A_s6s5 =
(angle_triangle_3A_s6s5*current_data(current_device,6)); % x
                    opp_triangle_3A_s6s5 =
sqrt((current_data(current_device,6))^2-(adj_triangle_3A_s6s5)^2); % y
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 3.1;
                    results_triangulation(current_row,4) = 11.195-
adj_triangle_3A_s6s5; % Need to subtract adj from width to obtain same x/y-axis,
then put adj value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
opp_triangle_3A_s6s5; % Need to subtract opp from length to obtain same x/y-
axis, then put opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
       % 3 Triangle B perspective (s5-s6)
```

```
            if ((current_data(current_device,5)+current_data(current_device,6) >
11.195) && (current_data(current_device,5)+current_data(current_device,6) <
30.7) && ((current_data(current_device,5) < 15.35) &&
(current_data(current_device,6) < 15.35)))
            angle_triangle_3B_s5s6 =
((current_data(current_device,5))^2+(current_data(current_device,6))^2-
(11.195)^2)/(2*(current_data(current_device,5)*current_data(current_device,6)));
                if (angle_triangle_3B_s5s6 < 1 && angle_triangle_3B_s5s6
>cosd(65))
                    adj_triangle_3B_s5s6 =
(angle_triangle_3B_s5s6*current_data(current_device,5)); % x
                    opp_triangle_3B_s5s6 =
sqrt((current_data(current_device,5))^2-(adj_triangle_3B_s5s6)^2); % y
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 3.2;
                    results_triangulation(current_row,4) = adj_triangle_3B_s5s6;
% Put adj value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
opp_triangle_3B_s5s6; % Need to subtract opp from length to obtain same x/y-
axis, then put opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 4 Triangle A perspective (s2-s6) Not so accurate
            if ((current_data(current_device,2)+current_data(current_device,6) >
20.8) && (current_data(current_device,2)+current_data(current_device,6) < 30.7)
&& ((current_data(current_device,2) < 15.35) && (current_data(current_device,6)
< 15.35)))
            angle_triangle_4A_s2s6 =
((20.80)^2+(current_data(current_device,2))^2-
(current_data(current_device,6))^2)/(2*20.80*(current_data(current_device,2)));
                if (angle_triangle_4A_s2s6 < 1 && angle_triangle_4A_s2s6
>=cosd(45))
                    adj_triangle_4A_s2s6 =
(angle_triangle_4A_s2s6*current_data(current_device,2)); % y
                    opp_triangle_4A_s2s6 =
sqrt((current_data(current_device,2))^2-(adj_triangle_4A_s2s6)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 4.1;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_4A_s2s6; % Need to subtract opp from width to obtain same x/y-axis,
then put adj value in fourth column as x
```

```
                    results_triangulation(current_row,5) = adj_triangle_4A_s2s6;
% Put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
          % 4 Triangle B perspective (s6-s2) Not so accurate
            if ((current_data(current_device,6)+current_data(current_device,2) >
20.8) && (current_data(current_device,6)+current_data(current_device,2) < 30.7)
&& ((current_data(current_device,6) < 15.35) && (current_data(current_device,2)
< 15.35)))
            angle_triangle_4B_s6s2 =
((current_data(current_device,6))^2+(current_data(current_device,2))^2-
(11.195)^2)/(2*(current_data(current_device,6)*current_data(current_device,2)));
                if (angle_triangle_4B_s6s2 < 1 && angle_triangle_4B_s6s2
>=cosd(45))
                    adj_triangle_4B_s6s2 =
(angle_triangle_4B_s6s2*current_data(current_device,6)); % y
                    opp_triangle_4B_s6s2 =
sqrt((current_data(current_device,6))^2-(adj_triangle_4B_s6s2)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 4.2;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_4B_s6s2; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_4B_s6s2; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 5 Triangle A perspective (s3-s1) (rotated)
            if ((current_data(current_device,3)+current_data(current_device,1) >
10.40) && (current_data(current_device,3)+current_data(current_device,1) < 15.7)
&& ((current_data(current_device,3) < 11.195) && (current_data(current_device,1)
< 11.195)))
            angle_triangle_5A_s3s1 =
((10.4)^2+(current_data(current_device,3))^2-
(current_data(current_device,1))^2)/(2*10.4*(current_data(current_device,3)));
                if (angle_triangle_5A_s3s1 < 1 && angle_triangle_5A_s3s1 >0)
                    adj_triangle_5A_s3s1 =
(angle_triangle_5A_s3s1*current_data(current_device,3)); % y
                    opp_triangle_5A_s3s1 =
sqrt((current_data(current_device,3))^2-(adj_triangle_5A_s3s1)^2); % x
```

```
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 5.1;
                    results_triangulation(current_row,4) = opp_triangle_5A_s3s1;
% Put opp value in fourth column as x (triangle is rotated)
                    results_triangulation(current_row,5) = 10.40-
adj_triangle_5A_s3s1; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y (triangle is rotated)
                    current_row = current_row +1;
                else
                end
            end
         % 5 Triangle B perspective (s1-s3)
            if ((current_data(current_device,1)+current_data(current_device,3) >
10.40) && (current_data(current_device,1)+current_data(current_device,3) < 15.7)
&& ((current_data(current_device,1) < 11.195) && (current_data(current_device,3)
< 11.195)))
            angle_triangle_5B_s1s3 =
((current_data(current_device,1))^2+(current_data(current_device,3))^2-
(11.195)^2)/(2*(current_data(current_device,1)*current_data(current_device,3)));
                if (angle_triangle_5B_s1s3 < 1 && angle_triangle_5B_s1s3 >0)
                    adj_triangle_5B_s1s3 =
(angle_triangle_5B_s1s3*current_data(current_device,1)); % y
                    opp_triangle_5B_s1s3 =
sqrt((current_data(current_device,1))^2-(adj_triangle_5B_s1s3)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 5.2;
                    results_triangulation(current_row,4) = opp_triangle_5B_s1s3;
% Put opp value in fourth column as x (triangle is rotated)
                    results_triangulation(current_row,5) = adj_triangle_5B_s1s3;
% Put adj value in fifth column as y (triangle is rotated)
                    current_row = current_row +1;
                else
                end
            end
          % 6 Triangle A perspective (s5-s3)
            if ((current_data(current_device,5)+current_data(current_device,3) >
10.40) && (current_data(current_device,5)+current_data(current_device,3) < 15.7)
&& ((current_data(current_device,5) < 11.195) && (current_data(current_device,3)
< 11.195)))
            angle_triangle_6A_s5s3 =
((10.4)^2+(current_data(current_device,5))^2-
(current_data(current_device,3))^2)/(2*10.4*(current_data(current_device,5)));
                if (angle_triangle_6A_s5s3 < 1 && angle_triangle_6A_s5s3 >0)
```

```
                    adj_triangle_6A_s5s3 =
(angle_triangle_6A_s5s3*current_data(current_device,5)); % y
                    opp_triangle_6A_s5s3 =
sqrt((current_data(current_device,5))^2-(adj_triangle_6A_s5s3)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 6.1;
                    results_triangulation(current_row,4) = opp_triangle_6A_s5s3;
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_6A_s5s3; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 6 Triangle B perspective (s3-s5)
            if ((current_data(current_device,3)+current_data(current_device,5) >
10.40) && (current_data(current_device,3)+current_data(current_device,5) < 15.7)
&& ((current_data(current_device,3) < 11.195) && (current_data(current_device,5)
< 11.195)))
            angle_triangle_6B_s3s5 =
((current_data(current_device,3))^2+(current_data(current_device,5))^2-
(11.195)^2)/(2*(current_data(current_device,3)*current_data(current_device,5)));
                if (angle_triangle_6B_s3s5 < 1 && angle_triangle_6B_s3s5 >0)
                    adj_triangle_6B_s3s5 =
(angle_triangle_6B_s3s5*current_data(current_device,3)); % y
                    opp_triangle_6B_s3s5 =
sqrt((current_data(current_device,3))^2-(adj_triangle_6B_s3s5)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 6.2;
                    results_triangulation(current_row,4) = opp_triangle_6B_s3s5;
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) =
10.40+adj_triangle_6B_s3s5; % Need to add length to adj to obtain same x/y-axis,
then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
           % 7 Triangle A perspective (s4-s6)
            if ((current_data(current_device,4)+current_data(current_device,6) >
10.40) && (current_data(current_device,4)+current_data(current_device,6) < 15.7)
&& ((current_data(current_device,4) < 11.195) && (current_data(current_device,6)
< 11.195)))
```

```
            angle_triangle_7A_s4s6 = 
((10.4)^2+(current_data(current_device,4))^2-
(current_data(current_device,6))^2)/(2*10.4*(current_data(current_device,4)));
                if (angle_triangle_7A_s4s6 < 1 && angle_triangle_7A_s4s6 >0)
                    adj_triangle_7A_s4s6 = 
(angle_triangle_7A_s4s6*current_data(current_device,4)); % y
                    opp_triangle_7A_s4s6 = 
sqrt((current_data(current_device,4))^2-(adj_triangle_7A_s4s6)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 7.1;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_7A_s4s6; % Need to subtract opp from width to obtain same x/y-axis, 
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = 
10.40+adj_triangle_7A_s4s6; % Need to add length to adj to obtain same x/y-axis, 
then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
            % 7 Triangle B perspective (s6-s4)
            if ((current_data(current_device,6)+current_data(current_device,4) > 
10.40) && (current_data(current_device,6)+current_data(current_device,4) < 15.7) 
&& ((current_data(current_device,6) < 11.195) && (current_data(current_device,4) 
< 11.195)))
            angle_triangle_7B_s6s4 = 
((current_data(current_device,6))^2+(current_data(current_device,4))^2-
(11.195)^2)/(2*(current_data(current_device,6)*current_data(current_device,4)));
                if (angle_triangle_7B_s6s4 < 1 && angle_triangle_7B_s6s4 >0)
                    adj_triangle_7B_s6s4 = 
(angle_triangle_7B_s6s4*current_data(current_device,6)); % y
                    opp_triangle_7B_s6s4 = 
sqrt((current_data(current_device,6))^2-(adj_triangle_7B_s6s4)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 7.2;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_7B_s6s4; % Need to subtract opp from width to obtain same x/y-axis, 
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_7B_s6s4; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
          % 8 Triangle A perspective (s2-s4)
            if ((current_data(current_device,2)+current_data(current_device,4) >
10.40) && (current_data(current_device,2)+current_data(current_device,4) < 15.7)
&& ((current_data(current_device,2) < 11.195) && (current_data(current_device,4)
< 11.195)))
            angle_triangle_8A_s2s4 =
((10.4)^2+(current_data(current_device,2))^2-
(current_data(current_device,4))^2)/(2*10.4*(current_data(current_device,2)));
                if (angle_triangle_8A_s2s4 < 1 && angle_triangle_8A_s2s4 >0)
                    adj_triangle_8A_s2s4 =
(angle_triangle_8A_s2s4*current_data(current_device,2)); % Need to calculate y
first (the triangle is rotated).
                    opp_triangle_8A_s2s4 =
sqrt((current_data(current_device,2))^2-(adj_triangle_8A_s2s4)^2);
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 8.1;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_8A_s2s4; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = adj_triangle_8A_s2s4;
% Put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 8 Triangle B perspective (s4-s2)
            if ((current_data(current_device,4)+current_data(current_device,2) >
10.40) && (current_data(current_device,4)+current_data(current_device,2) < 15.7)
&& ((current_data(current_device,4) < 11.195) && (current_data(current_device,2)
< 11.195)))
            angle_triangle_8B_s4s2 =
((current_data(current_device,4))^2+(current_data(current_device,2))^2-
(11.195)^2)/(2*(current_data(current_device,4)*current_data(current_device,2)));
                if (angle_triangle_8B_s4s2 < 1 && angle_triangle_8B_s4s2 >0)
                    adj_triangle_8B_s4s2 =
(angle_triangle_8B_s4s2*current_data(current_device,4)); % y
                    opp_triangle_8B_s4s2 =
sqrt((current_data(current_device,4))^2-(adj_triangle_8B_s4s2)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_device;
                    results_triangulation(current_row,3) = 8.2;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_8B_s4s2; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
```

```
                    results_triangulation(current_row,5) = 10.40-
adj_triangle_8B_s4s2; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
           % Remove negative values in results_triangulation and results outside
of room
           results_triangulation =
results_triangulation(results_triangulation(:,4)>0,:);
           results_triangulation =
results_triangulation(results_triangulation(:,5)>0,:);
           results_triangulation =
results_triangulation(results_triangulation(:,4)<11.195,:);
           results_triangulation =
results_triangulation(results_triangulation(:,5)<20.80,:);
           current_row = size(find(results_triangulation(:,1)>0),1)+1; % Find
total rows after removing negative values
           % Pre-work If results greater than 0, calculate mean value
(coordinate) among all triangles. Do it for both Wifi and Blueototh data
           currentresults =
results_triangulation((results_triangulation(:,1)==technologies) &
(results_triangulation(:,2)==current_device),:);  % Save results that only
contain current device and current technology
                 if size(currentresults)>=1 % Test to see if data is zero
                     if technologies == 2 % Test to see if wifi or bluetooth
                       xy_results_triangulation_bluetooth_currentlocation =
currentresults(:,3:5); % Use x and y value in column 4 and 5
                       % Bluetooth - If there exists multiple results for the
same second, calculate mean value of these coordinates
                       count_current_values =
size(xy_results_triangulation_bluetooth_currentlocation(:,1));
                         if count_current_values(1) >= 1 % if there exists
multiple results, then statement is true

results_triangulation_bluetooth_mean(current_device,1) = currentresults(1,2); %
put device number in first column

results_triangulation_bluetooth_mean(current_device,2) =
mean(xy_results_triangulation_bluetooth_currentlocation(:,2)); % mean of x
values

results_triangulation_bluetooth_mean(current_device,3) =
mean(xy_results_triangulation_bluetooth_currentlocation(:,3)); % mean of y
values
```

```
                             results_triangulation_bluetooth_mean =
results_triangulation_bluetooth_mean(results_triangulation_bluetooth_mean(:,1)>0
,:); % remove null values
                         end
                     else
                       xy_results_triangulation_wifi_currentlocation =
currentresults(:,3:5);
                       % WiFi - If there exists multiple results for the same
second, calculate mean value of these coordinates
                       count_current_values =
size(xy_results_triangulation_wifi_currentlocation(:,1));
                         if count_current_values(1) >= 1 % if there exists
multiple results, then statement is true
                             results_triangulation_wifi_mean(current_device,1) =
currentresults(1,2); % put device number in first column
                             results_triangulation_wifi_mean(current_device,2) =
mean(xy_results_triangulation_wifi_currentlocation(:,2)); % mean of x values
                             results_triangulation_wifi_mean(current_device,3) =
mean(xy_results_triangulation_wifi_currentlocation(:,3)); % mean of y values
                             results_triangulation_wifi_mean =
results_triangulation_wifi_mean(results_triangulation_wifi_mean(:,1)>0,:); %
remove null values
                         end
                     end
                 else
                 end
         end
         current_data = Results_static_bluetooth_distance; % Changing technology
and data set
     end

      % Pre-work for plotting the xy-coordinates
       for d=1:size(results_triangulation(:,1)) % Could the sum of x and y help
determine where in the room the device is located?

results_triangulation(d,6)=results_triangulation(d,4)+results_triangulation(d,5)
; % Sum x and y in column 6
       end

       for d=1:size(Results_static_bluetooth_distance(:,1)) % Just to see if I
can determine if the device in on the left/right side of the room

Results_static_bluetooth_distance(d,7)=Results_static_bluetooth_distance(d,1)+Re
sults_static_bluetooth_distance(d,3)+Results_static_bluetooth_distance(d,5); %
Sum distances from sensor 1,3,5 in column 7
```

```
Results_static_bluetooth_distance(d,8)=Results_static_bluetooth_distance(d,2)+Re
sults_static_bluetooth_distance(d,4)+Results_static_bluetooth_distance(d,6); %
Sum distances from sensor 2,4,6 in column 8
       end

       wifilength = size(find(results_triangulation(:,1)==1));
       results_triangulation_wifi = results_triangulation(1:wifilength,:);
       bluetoothlength = size(find(results_triangulation(:,1)==2));
       results_triangulation_bluetooth =
results_triangulation(wifilength+1:(wifilength+bluetoothlength),:);

       %     Plotting Wi-Fi, Bluetooth xy-coordinates, sensors from method and
static locations. Comment out code to choose between technology and all/mean
results
             figure();
             set(gcf, 'Position',  [100, 100, 550, 650]);

plot(sensors_triangulation(1,:),sensors_triangulation(2,:),'rs','MarkerSize',
12, 'LineWidth',2,'DisplayName','Actual position');

text(sensors_triangulation(1,:)+0.2,sensors_triangulation(2,:)+0.2,{'s1','s2','s
3','s4','s5','s6'},'FontSize',14);
             hold on;
             for l=1:10
             %    if l==10

scatter(locations(l,1),locations(l,2),50,'filled','green','square'); % Plotting
all 10 locations
             text(locations(l,1),locations(l,2),sprintf(' %.f',l),'FontSize',
14)
              %   else
               %  end
             end
             % Wifi
             for d=1:length(results_triangulation_wifi_mean(:,1)) % Wifi mean
               % if d==10
                 counter = results_triangulation_wifi_mean(d,1);

scatter(results_triangulation_wifi_mean(d,2),results_triangulation_wifi_mean(d,3
),40,'filled','^','MarkerFaceColor',base_color(counter,:));

text(results_triangulation_wifi_mean(d,2),results_triangulation_wifi_mean(d,3),s
printf(' %.fW',counter),'FontSize', 14);
                %else
                %end
             end
             % Bluetooth
             for d=1:length(results_triangulation_bluetooth_mean(:,1)) %
Bluetooth mean
                 %if d==10
                 counter = results_triangulation_bluetooth_mean(d,1);

scatter(results_triangulation_bluetooth_mean(d,2),results_triangulation_bluetoot
h_mean(d,3),40,'filled','^','MarkerFaceColor',base_color(counter,:));

text(results_triangulation_bluetooth_mean(d,2),results_triangulation_bluetooth_m
ean(d,3),sprintf(' %.fB',counter),'FontSize', 14);
                 %else
                 %end
             end
            % title('Results from triangulation - Static'); % Give name to the
plot
             axis([0 11.195 0 20.8]);
             xticks(0:1:11.195);
             yticks(0:1:20.80);
             grid on;

      % Finding Euclidean distance from known location to Wi-Fi and Bluetooth
data  - Air distance
     for t=1:2
         for l=1:10
            if t==1
                air_distance(l,2) =
pdist([results_triangulation_wifi_mean(l,2:3);locations(l,:)],'euclidean'); %
format: d = pdist(coordinates,'euclidean')
            else
                air_distance(l,3) =
pdist([results_triangulation_bluetooth_mean(l,2:3);locations(l,:)],'euclidean');
% calculate Euclidean distance
                air_distance(l,1) = l; % Put device number in first column
            end
         end
     end
     % Setting variable names to the table and converts to table
      air_distance = array2table(air_distance,...
            'VariableNames',{'Device #' 'Wi-Fi [m]' 'Bluetooth [m]'});

     elseif method == 3
```

## Trilateration

```
    xy_results = zeros(10,4);
    xy_results1 = zeros(10,4);
    xy_results2 = zeros(10,4);
    xy_results3 = zeros(10,4);
    xy_results4 = zeros(10,4);
    air_distance = zeros(10,3);
    xy_results_trilateration_wifi_mean = zeros(10,3);
    xy_results_trilateration_bluetooth_mean = zeros(10,3);
    Data=Results_static_wifi_distance;

    for technologies = 1:2 % loop through both Wi-Fi and Bluetooth data
          length_of_data = 1;
        for length_of_data = 1:size(Data,1)

        % Stores values from sensor 1,2,5,6 based on current row into r1..r6
         r1 = (Data(length_of_data,1));
         r2 = (Data(length_of_data,2));
         r5 = (Data(length_of_data,5));
         r6 = (Data(length_of_data,6));

% Calculations explained --> dynamic --> trilateration

    A = 2*(-xy(1,1)+xy(1,2)); % 2(-x1+x2)
    B = 2*(-xy(2,1)+xy(2,2)); % 2(-y1+y2)
    C = (r1)^2-(r2)^2-(xy(1,1))^2+(xy(1,2))^2-(xy(2,1))^2+(xy(2,2))^2;

    D = 2*(-xy(1,2)+xy(1,3));
    E = 2*(-xy(2,2)+xy(2,3));
    F = (r2)^2-(r5)^2-(xy(1,2))^2+(xy(1,3))^2-(xy(2,2))^2+(xy(2,3))^2;

    G = 2*(-xy(1,3)+xy(1,4));
    H = 2*(-xy(2,3)+xy(2,4));
    I = (r5)^2-(r6)^2-(xy(1,3))^2+(xy(1,4))^2-(xy(2,3))^2+(xy(2,4))^2;

    J = 2*(-xy(1,4)+xy(1,1));
    K = 2*(-xy(2,4)+xy(2,1));
    L = (r6)^2-(r1)^2-(xy(1,4))^2+(xy(1,1))^2-(xy(2,4))^2+(xy(2,1))^2;

            % 1 Equation 1 (Sensor 1,2,5)
            x_1 = abs((C*E-B*F)/(A*E-B*D));
            y_1 = abs((C*D-A*F)/(B*D-A*E));

            % 2 Equation 2 (Sensor 2,5,6)
            x_2 = abs((F*H-E*I)/(D*H-E*G));
            y_2 = abs((F*G-D*I)/(E*G-D*H));
```

```
             % 3 Equation 3 (Sensor 5,6,1)
             x_3 = abs((I*K-H*L)/(G*K-H*J));
             y_3 = abs((I*J-G*L)/(H*J-G*K));

             % 4 Equation 4 (Sensor 6-1-2)
             x_4 = abs((L*A-K*C)/(J*B-K*A));
             y_4 = abs((L*A-J*C)/(K*A-J*B));

             % Save values of x and y into tables of results based on different
equations, remove values outside of room
             if x_1 > 11.195 || y_1 > 21.5
             else
                 xy_results1(length_of_data,1) = x_1; % Stores x values in table
xy_results1 from equation 1
                 xy_results1(length_of_data,2) = y_1; % Stores y values in table
xy_results1 from equation 1
                 xy_results1(length_of_data,3) = 1; % Store equation number in
column 3
                 xy_results1(length_of_data,4) = length_of_data; % Store device
number in column 4
             end
             if x_2 > 11.195 || y_2 > 21.5
             else
                 xy_results2(length_of_data,1) = x_2; % Stores x values in table
xy_results2 from equation 2
                 xy_results2(length_of_data,2) = y_2; % Stores y values in table
xy_results2 from equation 2
                 xy_results2(length_of_data,3) = 2; % Store equation number in
column 3
                 xy_results2(length_of_data,4) = length_of_data; % Store device
number in column 4
             end
             if x_3 > 11.195 || y_3 > 21.5
             else
                 xy_results3(length_of_data,1) = x_3; % Stores x values in table
xy_results3 from equation 3
                 xy_results3(length_of_data,2) = y_3; % Stores y values in table
xy_results3 from equation 3
                 xy_results3(length_of_data,3) = 3; % Store equation number in
column 3
                 xy_results3(length_of_data,4) = length_of_data; % Store device
number in column 4
             end
             if x_4 > 11.195 || y_4 > 21.5
             else
```

```
                 xy_results4(length_of_data,1) = x_4; % Stores x values in table 
xy_results4 from equation 4
                 xy_results4(length_of_data,2) = y_4; % Stores y values in table 
xy_results4 from equation 4
                 xy_results4(length_of_data,3) = 4; % Store equation number in 
column 3
                 xy_results4(length_of_data,4) = length_of_data; % Store device 
number in column 4
             end
             
                 % Finding mean value of xy_results1-4 related to static 
location
                 if technologies == 2
                   xy_results_trilateration_bluetooth_total = 
[xy_results1(length_of_data,:); xy_results2(length_of_data,:); 
xy_results3(length_of_data,:); xy_results4(length_of_data,:)];
                   xy_results_trilateration_bluetooth_total = 
xy_results_trilateration_bluetooth_total(xy_results_trilateration_bluetooth_tota
l(:,1)>0,:); % remove null values
                 % Bluetooth - If there exists multiple results for the same 
second, calculate mean value of these coordinates
                   count_current_values = 
size(xy_results_trilateration_bluetooth_total(:,1));
                     if count_current_values(1) >= 1 % if there exists multiple 
results, then statement is true
                         
xy_results_trilateration_bluetooth_mean(length_of_data,1) = 
xy_results_trilateration_bluetooth_total(1,4); % put device number in first 
column
                         
xy_results_trilateration_bluetooth_mean(length_of_data,2) = 
mean(xy_results_trilateration_bluetooth_total(:,1)); % mean of x values
                         
xy_results_trilateration_bluetooth_mean(length_of_data,3) = 
mean(xy_results_trilateration_bluetooth_total(:,2)); % mean of y values
                         xy_results_trilateration_bluetooth_mean = 
xy_results_trilateration_bluetooth_mean(xy_results_trilateration_bluetooth_mean(
:,1)>0,:); % remove null values
                     else
                         
xy_results_trilateration_bluetooth_mean(length_of_data,1) = length_of_data; % 
put device number in first column
                         
xy_results_trilateration_bluetooth_mean(length_of_data,2:3) = 0;
                         continue; % Passes control to the next iteration of 
lengthof_data is zero
```

```
                    end
                else
                  xy_results_trilateration_wifi_total =
[xy_results1(length_of_data,:); xy_results2(length_of_data,:);
xy_results3(length_of_data,:); xy_results4(length_of_data,:)];
                  xy_results_trilateration_wifi_total =
xy_results_trilateration_wifi_total(xy_results_trilateration_wifi_total(:,1)>0,:
); % remove null values
                  % WiFi - If there exists multiple results for the same
second, calculate mean value of these coordinates
                  count_current_values =
size(xy_results_trilateration_wifi_total(:,1));
                    if count_current_values(1) >= 1 % if there exists multiple
results, then statement is true
                        xy_results_trilateration_wifi_mean(length_of_data,1) =
xy_results_trilateration_wifi_total(1,4); % put device number in first column
                        xy_results_trilateration_wifi_mean(length_of_data,2) =
mean(xy_results_trilateration_wifi_total(:,1)); % mean of x values
                        xy_results_trilateration_wifi_mean(length_of_data,3) =
mean(xy_results_trilateration_wifi_total(:,2)); % mean of y values
                        xy_results_trilateration_wifi_mean =
xy_results_trilateration_wifi_mean(xy_results_trilateration_wifi_mean(:,1)>0,:);
% remove null values
                    else
                        xy_results_trilateration_wifi_mean(length_of_data,1) =
length_of_data; % put device number in first column
                        xy_results_trilateration_wifi_mean(length_of_data,2:3)
= 0;

                        continue; % Passes control to the next iteration of
lengthof_data is zero
                    end
                end
         end
             length_of_data = length_of_data + 1;

                % Save results from each equation into seperate variables
                if technologies == 2
                    xy_results_trilateration_bluetooth_1 =
xy_results1(xy_results1(:,1)>0,:); % Save only valid coordiantes (0,0 is
invalid)
                    xy_results_trilateration_bluetooth_2 =
xy_results2(xy_results2(:,1)>0,:);
                    xy_results_trilateration_bluetooth_3 =
xy_results3(xy_results3(:,1)>0,:);
                    xy_results_trilateration_bluetooth_4 =
xy_results4(xy_results4(:,1)>0,:);
```

```
                 else
                     Data=Results_static_bluetooth_distance;
                     xy_results_trilateration_wifi_1 =
xy_results1(xy_results1(:,1)>0,:);
                     xy_results_trilateration_wifi_2 =
xy_results2(xy_results2(:,1)>0,:);
                     xy_results_trilateration_wifi_3 =
xy_results3(xy_results3(:,1)>0,:);
                     xy_results_trilateration_wifi_4 =
xy_results4(xy_results4(:,1)>0,:);
                     xy_results1 = zeros(10,4);    % resetting before bluetooth
technology
                     xy_results2 = zeros(10,4);
                     xy_results3 = zeros(10,4);
                     xy_results4 = zeros(10,4);
                 end
     end
     % Concatenating all bluetooth and wifi results into two tables
     xy_results_trilateration_wifi_total =
cat(1,xy_results_trilateration_wifi_1,xy_results_trilateration_wifi_2,xy_results
_trilateration_wifi_3,xy_results_trilateration_wifi_4);
     xy_results_trilateration_bluetooth_total =
cat(1,xy_results_trilateration_bluetooth_1,xy_results_trilateration_bluetooth_2,
xy_results_trilateration_bluetooth_3,xy_results_trilateration_bluetooth_4);

     %     Plotting all/mean wifi and blutooth location (+ Sensors and static
locations)
             figure();
             set(gcf, 'Position',  [100, 100, 600, 600]);
             plot(xy(1,:),xy(2,:),'rs','MarkerSize', 12,
'LineWidth',2,'DisplayName','Actual position');
             text(xy(1,:),xy(2,:),{'s1','s2','s5','s6'});
             hold on;
             for d=1:10 % Current static locations
                %if d==10

scatter(locations(d,1),locations(d,2),40,'filled','green','square'); % plot
marker
                 text(locations(d,1),locations(d,2),sprintf('
%.f',d),'FontSize', 14) % plot text related to marker
             end
             % Wi-Fi
             for d=1:length(xy_results_trilateration_wifi_mean(:,1)) % Mean wifi
device location
                 counter = xy_results_trilateration_wifi_mean(d,1); % Use
counter to get right color on marker
```

```
scatter(xy_results_trilateration_wifi_mean(d,2),xy_results_trilateration_wifi_me
an(d,3),40,'filled','MarkerFaceColor',base_color(counter,:));

text(xy_results_trilateration_wifi_mean(d,2),xy_results_trilateration_wifi_mean(
d,3),sprintf(' W%.f',xy_results_trilateration_wifi_mean(d,1)),'FontSize',14);
             end

             % Bluetooth
             for d=1:length(xy_results_trilateration_bluetooth_mean(:,1)) % Mean
wifi device location
                 counter = xy_results_trilateration_bluetooth_mean(d,1); % Use
counter to get right color on marker

scatter(xy_results_trilateration_bluetooth_mean(d,2),xy_results_trilateration_bl
uetooth_mean(d,3),40,'filled','MarkerFaceColor',base_color(counter,:));

text(xy_results_trilateration_bluetooth_mean(d,2),xy_results_trilateration_bluet
ooth_mean(d,3),sprintf('
B%.f',xy_results_trilateration_bluetooth_mean(d,1)),'FontSize',14);
             end
             %title('Device locations using trilateration'); % Give name to the
plot
             axis([0 11.195 0 20.8]);
             xticks(0:1:11.195);
             grid on;

  % Finding Euclidean distance from known location to Wi-Fi and Bluetooth data -
Air distance
     for t=1:2
         for l=1:10
            if t==1
                air_distance(l,2) =
pdist([xy_results_trilateration_wifi_mean(l,2:3);locations(l,:)],'euclidean'); %
format: d = pdist(coordinates,'euclidean')
            else
                air_distance(l,3) =
pdist([xy_results_trilateration_bluetooth_mean(l,2:3);locations(l,:)],'euclidean
'); % calculate Euclidean distance
                air_distance(l,1) = l; % Put device number in first column
            end
         end
     end
     % Setting variable names to the table and converts to table
      air_distance = array2table(air_distance,...
            'VariableNames',{'Device #' 'Wi-Fi [m]' 'Bluetooth [m]'});
```

```
    end

else % Choosing dynamic results based on input values
```

## DYNAMIC

```
if method == 2
```

## Triangulation

```
      results_triangulation = zeros(500,5);
      current_data = Results_dynamic_wifi_distance; % Choosing the wifi data set
      current_row = 1;
     for technologies = 1:2 % loop through both Wi-Fi and Bluetooth data
         current_time = 1; % Starting at timestamp one
         for current_time = 1:size(current_data,1) % Looping through all
timestamps, starting at one
                       % 1 Triangle A perspective (s1-s2)
             % Check for "true" triangles: Check if distance between s1-s2 >
widt and distance between s5-s1 > length, also that the signal converted to
distance is shorter than 22.5 and that both distances does not exeeds 45
             if ((current_data(current_time,1)+current_data(current_time,2) >
11.195) && (current_data(current_time,1)+current_data(current_time,2) < 30.7) &&
((current_data(current_time,1) < 15.35) && (current_data(current_time,2) <
15.35)))
             % Step 1 - Calculate angle in triangle 1A with the law of cosines:
a^2 = b2^2 + c2^2-2bc*Cos(A) --> Cos(A) = (b2^2+c^2-a^2)/(2*bc)
                    angle_triangle_1A_s1s2 =
((11.195)^2+(current_data(current_time,1))^2-
(current_data(current_time,2))^2)/(2*11.195*(current_data(current_time,1))); %
Law of cosines
             % Step 2 - Calculate the (adjacent) in a right-angled triangle with
the law of cosines: cos(A) = adjacent / hypotenuse, where hyp = a --> cos(A) * a
= adj
                if (angle_triangle_1A_s1s2 < 1 && angle_triangle_1A_s1s2
>cosd(62.5)) % Check angle - Must be between 0 and 45 degrees, filtering out
angles outside of the sector/room
                    adj_triangle_1A_s1s2 =
(angle_triangle_1A_s1s2*current_data(current_time,1)); % adj = angle * c
             % Step 3 - Calculate opposite with pytagoras theorem: c^2 = adj^2 +
opp^2
                    opp_triangle_1A_s1s2 =
sqrt((current_data(current_time,1))^2-(adj_triangle_1A_s1s2)^2); %
             % Storing opp/adj-values in triangulation_results --> (adj,opp)
                    results_triangulation(current_row,1) = technologies; % Put
current device in first column
```

```
                    results_triangulation(current_row,2) = current_time;  % Put
current technology in second column
                    results_triangulation(current_row,3) = 1.1; % Put current
triangle in third column
                    results_triangulation(current_row,4) = adj_triangle_1A_s1s2;
% Put current adj value in fourth column as x
                    results_triangulation(current_row,5) = opp_triangle_1A_s1s2;
% Put current opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
             end
                 % 1 Triangle B perspective (s2-s1)
             % Check for "true" triangles: Check if distance between s1-s2 >
widt and distance between s5-s1 > length, also that the signal converted to
distance is shorter than 22.5 and that both distances does not exeeds 45
             if ((current_data(current_time,2)+current_data(current_time,1) >
11.195) && (current_data(current_time,2)+current_data(current_time,1) < 30.7) &&
((current_data(current_time,2) < 15.35) && (current_data(current_time,1) <
15.35)))
             % Step 1 - Calculate angle in triangle 1B with the law of cosines:
b^2 = c^2+a^2-2ca*Cos(B) --> Cos(B) = (c^2+a^2-b^2)/2ca
                    angle_triangle_1B_s2s1 =
((current_data(current_time,2))^2+(current_data(current_time,1))^2-
(11.195)^2)/(2*(current_data(current_time,2)*current_data(current_time,1))); %
Law of cosines
             % Step 2 - Calculate the (adjacent) in a right-angled triangle with
the law of cosines: cos(A) = adjacent / hypotenuse, where hyp = a --> cos(A) * a
= adj
                if (angle_triangle_1B_s2s1 < 1 && angle_triangle_1B_s2s1
>cosd(62.5)) % Check angle - Must be between 0 and 1, filtering out angles
outside of the room
                    adj_triangle_1B_s2s1 =
(angle_triangle_1B_s2s1*current_data(current_time,2)); % adj = angle * a
             % Step 3 - Calculate opposite with pytagoras theorem: a^2 = adj^2 +
opp^2
                    opp_triangle_1B_s2s1 =
sqrt((current_data(current_time,2))^2-(adj_triangle_1B_s2s1)^2); %
             % Storing opp/adj-values in triangulation_results --> (adj,opp)
                    results_triangulation(current_row,1) = technologies; % Put
current device in first column
                    results_triangulation(current_row,2) = current_time;  % Put
current technology in second column
                    results_triangulation(current_row,3) = 1.2; % Put current
triangle in third column
```

```
                    results_triangulation(current_row,4) = 11.195-
adj_triangle_1B_s2s1; % Put current adj value in fourth column as x
                    results_triangulation(current_row,5) = opp_triangle_1B_s2s1;
% Put current opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 2 Triangle A perspective (s5-s1)
            if ((current_data(current_time,5)+current_data(current_time,1) >
20.8) && (current_data(current_time,5)+current_data(current_time,1) < 30.7) &&
((current_data(current_time,5) < 15.35) && (current_data(current_time,1) <
15.35)))
            angle_triangle_2A_s5s1 =
((20.80)^2+(current_data(current_time,5))^2-
(current_data(current_time,1))^2)/(2*20.80*(current_data(current_time,5)));
                if (angle_triangle_2A_s5s1 < 1 && angle_triangle_2A_s5s1
>=cosd(45))
                    adj_triangle_2A_s5s1 =
(angle_triangle_2A_s5s1*current_data(current_time,5)); % y
                    opp_triangle_2A_s5s1 =
sqrt((current_data(current_time,5))^2-(adj_triangle_2A_s5s1)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 2.1;
                    results_triangulation(current_row,4) = opp_triangle_2A_s5s1;
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_2A_s5s1; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 2 Triangle B perspective (s1-s5) - Most inaccurate
            if ((current_data(current_time,1)+current_data(current_time,5) >
20.8) && (current_data(current_time,1)+current_data(current_time,5) < 30.7) &&
((current_data(current_time,1) < 15.35) && (current_data(current_time,5) <
15.35)))
            angle_triangle_2B_s1s5 =
((current_data(current_time,1))^2+(current_data(current_time,5))^2-
(11.195)^2)/(2*(current_data(current_time,1)*current_data(current_time,5)));
                if (angle_triangle_2B_s1s5 < 1 && angle_triangle_2B_s1s5
>=cosd(45))
                    adj_triangle_2B_s1s5 =
(angle_triangle_2B_s1s5*current_data(current_time,1)); % y
```

```
                    opp_triangle_2B_s1s5 =
sqrt((current_data(current_time,1))^2-(adj_triangle_2B_s1s5)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 2.2;
                    results_triangulation(current_row,4) = opp_triangle_2B_s1s5;
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) = adj_triangle_2B_s1s5;
% Put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 3 Triangle A perspective (s6-s5)
            if ((current_data(current_time,6)+current_data(current_time,5) >
11.195) && (current_data(current_time,6)+current_data(current_time,5) < 30.7) &&
((current_data(current_time,6) < 15.35) && (current_data(current_time,5) <
15.35)))
            angle_triangle_3A_s6s5 =
((11.195)^2+(current_data(current_time,6))^2-
(current_data(current_time,5))^2)/(2*11.195*(current_data(current_time,6)));
                if (angle_triangle_3A_s6s5 < 1 && angle_triangle_3A_s6s5
>cosd(65))
                    adj_triangle_3A_s6s5 =
(angle_triangle_3A_s6s5*current_data(current_time,6)); % x
                    opp_triangle_3A_s6s5 =
sqrt((current_data(current_time,6))^2-(adj_triangle_3A_s6s5)^2); % y
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 3.1;
                    results_triangulation(current_row,4) = 11.195-
adj_triangle_3A_s6s5; % Need to subtract adj from width to obtain same x/y-axis,
then put adj value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
opp_triangle_3A_s6s5; % Need to subtract opp from length to obtain same x/y-
axis, then put opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
       % 3 Triangle B perspective (s5-s6)
            if ((current_data(current_time,5)+current_data(current_time,6) >
11.195) && (current_data(current_time,5)+current_data(current_time,6) < 30.7) &&
((current_data(current_time,5) < 15.35) && (current_data(current_time,6) <
15.35)))
```

```
            angle_triangle_3B_s5s6 =
((current_data(current_time,5))^2+(current_data(current_time,6))^2-
(11.195)^2)/(2*(current_data(current_time,5)*current_data(current_time,6)));
                if (angle_triangle_3B_s5s6 < 1 && angle_triangle_3B_s5s6
>cosd(65))
                    adj_triangle_3B_s5s6 =
(angle_triangle_3B_s5s6*current_data(current_time,5)); % x
                    opp_triangle_3B_s5s6 =
sqrt((current_data(current_time,5))^2-(adj_triangle_3B_s5s6)^2); % y
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 3.2;
                    results_triangulation(current_row,4) = adj_triangle_3B_s5s6;
% Put adj value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
opp_triangle_3B_s5s6; % Need to subtract opp from length to obtain same x/y-
axis, then put opp value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 4 Triangle A perspective (s2-s6) Not so accurate
            if ((current_data(current_time,2)+current_data(current_time,6) >
20.8) && (current_data(current_time,2)+current_data(current_time,6) < 30.7) &&
((current_data(current_time,2) < 15.35) && (current_data(current_time,6) <
15.35)))
            angle_triangle_4A_s2s6 =
((20.80)^2+(current_data(current_time,2))^2-
(current_data(current_time,6))^2)/(2*20.80*(current_data(current_time,2)));
                if (angle_triangle_4A_s2s6 < 1 && angle_triangle_4A_s2s6
>=cosd(45))
                    adj_triangle_4A_s2s6 =
(angle_triangle_4A_s2s6*current_data(current_time,2)); % y
                    opp_triangle_4A_s2s6 =
sqrt((current_data(current_time,2))^2-(adj_triangle_4A_s2s6)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 4.1;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_4A_s2s6; % Need to subtract opp from width to obtain same x/y-axis,
then put adj value in fourth column as x
                    results_triangulation(current_row,5) = adj_triangle_4A_s2s6;
% Put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
          % 4 Triangle B perspective (s6-s2) Not so accurate
            if ((current_data(current_time,6)+current_data(current_time,2) >
20.8) && (current_data(current_time,6)+current_data(current_time,2) < 30.7) &&
((current_data(current_time,6) < 15.35) && (current_data(current_time,2) <
15.35)))
            angle_triangle_4B_s6s2 =
((current_data(current_time,6))^2+(current_data(current_time,2))^2-
(11.195)^2)/(2*(current_data(current_time,6)*current_data(current_time,2)));
                if (angle_triangle_4B_s6s2 < 1 && angle_triangle_4B_s6s2
>=cosd(45))
                    adj_triangle_4B_s6s2 =
(angle_triangle_4B_s6s2*current_data(current_time,6)); % y
                    opp_triangle_4B_s6s2 =
sqrt((current_data(current_time,6))^2-(adj_triangle_4B_s6s2)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 4.2;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_4B_s6s2; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_4B_s6s2; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 5 Triangle A perspective (s3-s1) (rotated)
            if ((current_data(current_time,3)+current_data(current_time,1) >
10.40) && (current_data(current_time,3)+current_data(current_time,1) < 15.7) &&
((current_data(current_time,3) < 11.195) && (current_data(current_time,1) <
11.195)))
            angle_triangle_5A_s3s1 = ((10.4)^2+(current_data(current_time,3))^2-
(current_data(current_time,1))^2)/(2*10.4*(current_data(current_time,3)));
                if (angle_triangle_5A_s3s1 < 1 && angle_triangle_5A_s3s1 >0)
                    adj_triangle_5A_s3s1 =
(angle_triangle_5A_s3s1*current_data(current_time,3)); % y
                    opp_triangle_5A_s3s1 =
sqrt((current_data(current_time,3))^2-(adj_triangle_5A_s3s1)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 5.1;
                    results_triangulation(current_row,4) = opp_triangle_5A_s3s1;
% Put opp value in fourth column as x (triangle is rotated)
```

```
                    results_triangulation(current_row,5) = 10.40-
adj_triangle_5A_s3s1; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y (triangle is rotated)
                    current_row = current_row +1;
                else
                end
            end
         % 5 Triangle B perspective (s1-s3)
            if ((current_data(current_time,1)+current_data(current_time,3) >
10.40) && (current_data(current_time,1)+current_data(current_time,3) < 15.7) &&
((current_data(current_time,1) < 11.195) && (current_data(current_time,3) <
11.195)))
            angle_triangle_5B_s1s3 =
((current_data(current_time,1))^2+(current_data(current_time,3))^2-
(11.195)^2)/(2*(current_data(current_time,1)*current_data(current_time,3)));
                if (angle_triangle_5B_s1s3 < 1 && angle_triangle_5B_s1s3 >0)
                    adj_triangle_5B_s1s3 =
(angle_triangle_5B_s1s3*current_data(current_time,1)); % y
                    opp_triangle_5B_s1s3 =
sqrt((current_data(current_time,1))^2-(adj_triangle_5B_s1s3)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 5.2;
                    results_triangulation(current_row,4) = opp_triangle_5B_s1s3;
% Put opp value in fourth column as x (triangle is rotated)
                    results_triangulation(current_row,5) = adj_triangle_5B_s1s3;
% Put adj value in fifth column as y (triangle is rotated)
                    current_row = current_row +1;
                else
                end
            end
          % 6 Triangle A perspective (s5-s3)
            if ((current_data(current_time,5)+current_data(current_time,3) >
10.40) && (current_data(current_time,5)+current_data(current_time,3) < 15.7) &&
((current_data(current_time,5) < 11.195) && (current_data(current_time,3) <
11.195)))
            angle_triangle_6A_s5s3 = ((10.4)^2+(current_data(current_time,5))^2-
(current_data(current_time,3))^2)/(2*10.4*(current_data(current_time,5)));
                if (angle_triangle_6A_s5s3 < 1 && angle_triangle_6A_s5s3 >0)
                    adj_triangle_6A_s5s3 =
(angle_triangle_6A_s5s3*current_data(current_time,5)); % y
                    opp_triangle_6A_s5s3 =
sqrt((current_data(current_time,5))^2-(adj_triangle_6A_s5s3)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 6.1;
                    results_triangulation(current_row,4) = opp_triangle_6A_s5s3;
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_6A_s5s3; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
         % 6 Triangle B perspective (s3-s5)
            if ((current_data(current_time,3)+current_data(current_time,5) >
10.40) && (current_data(current_time,3)+current_data(current_time,5) < 15.7) &&
((current_data(current_time,3) < 11.195) && (current_data(current_time,5) <
11.195)))
            angle_triangle_6B_s3s5 =
((current_data(current_time,3))^2+(current_data(current_time,5))^2-
(11.195)^2)/(2*(current_data(current_time,3)*current_data(current_time,5)));
                if (angle_triangle_6B_s3s5 < 1 && angle_triangle_6B_s3s5 >0)
                    adj_triangle_6B_s3s5 =
(angle_triangle_6B_s3s5*current_data(current_time,3)); % y
                    opp_triangle_6B_s3s5 =
sqrt((current_data(current_time,3))^2-(adj_triangle_6B_s3s5)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 6.2;
                    results_triangulation(current_row,4) = opp_triangle_6B_s3s5;
% Put opp value in fourth column as x
                    results_triangulation(current_row,5) =
10.40+adj_triangle_6B_s3s5; % Need to add length to adj to obtain same x/y-axis,
then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
           % 7 Triangle A perspective (s4-s6)
            if ((current_data(current_time,4)+current_data(current_time,6) >
10.40) && (current_data(current_time,4)+current_data(current_time,6) < 15.7) &&
((current_data(current_time,4) < 11.195) && (current_data(current_time,6) <
11.195)))
            angle_triangle_7A_s4s6 = ((10.4)^2+(current_data(current_time,4))^2-
(current_data(current_time,6))^2)/(2*10.4*(current_data(current_time,4)));
                if (angle_triangle_7A_s4s6 < 1 && angle_triangle_7A_s4s6 >0)
                    adj_triangle_7A_s4s6 =
(angle_triangle_7A_s4s6*current_data(current_time,4)); % y
                    opp_triangle_7A_s4s6 =
sqrt((current_data(current_time,4))^2-(adj_triangle_7A_s4s6)^2); % x
```

```
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 7.1;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_7A_s4s6; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) =
10.40+adj_triangle_7A_s4s6; % Need to add length to adj to obtain same x/y-axis,
then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
            % 7 Triangle B perspective (s6-s4)
            if ((current_data(current_time,6)+current_data(current_time,4) >
10.40) && (current_data(current_time,6)+current_data(current_time,4) < 15.7) &&
((current_data(current_time,6) < 11.195) && (current_data(current_time,4) <
11.195)))
            angle_triangle_7B_s6s4 =
((current_data(current_time,6))^2+(current_data(current_time,4))^2-
(11.195)^2)/(2*(current_data(current_time,6)*current_data(current_time,4)));
                if (angle_triangle_7B_s6s4 < 1 && angle_triangle_7B_s6s4 >0)
                    adj_triangle_7B_s6s4 =
(angle_triangle_7B_s6s4*current_data(current_time,6)); % y
                    opp_triangle_7B_s6s4 =
sqrt((current_data(current_time,6))^2-(adj_triangle_7B_s6s4)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 7.2;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_7B_s6s4; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = 20.80-
adj_triangle_7B_s6s4; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
          % 8 Triangle A perspective (s2-s4)
            if ((current_data(current_time,2)+current_data(current_time,4) >
10.40) && (current_data(current_time,2)+current_data(current_time,4) < 15.7) &&
((current_data(current_time,2) < 11.195) && (current_data(current_time,4) <
11.195)))
            angle_triangle_8A_s2s4 = ((10.4)^2+(current_data(current_time,2))^2-
(current_data(current_time,4))^2)/(2*10.4*(current_data(current_time,2)));
```

```
                if (angle_triangle_8A_s2s4 < 1 && angle_triangle_8A_s2s4 >0)
                    adj_triangle_8A_s2s4 =
(angle_triangle_8A_s2s4*current_data(current_time,2)); % Need to calculate y
first (the triangle is rotated).
                    opp_triangle_8A_s2s4 =
sqrt((current_data(current_time,2))^2-(adj_triangle_8A_s2s4)^2);
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 8.1;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_8A_s2s4; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = adj_triangle_8A_s2s4;
% Put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
        % 8 Triangle B perspective (s4-s2)
            if ((current_data(current_time,4)+current_data(current_time,2) >
10.40) && (current_data(current_time,4)+current_data(current_time,2) < 15.7) &&
((current_data(current_time,4) < 11.195) && (current_data(current_time,2) <
11.195)))
            angle_triangle_8B_s4s2 =
((current_data(current_time,4))^2+(current_data(current_time,2))^2-
(11.195)^2)/(2*(current_data(current_time,4)*current_data(current_time,2)));
                if (angle_triangle_8B_s4s2 < 1 && angle_triangle_8B_s4s2 >0)
                    adj_triangle_8B_s4s2 =
(angle_triangle_8B_s4s2*current_data(current_time,4)); % y
                    opp_triangle_8B_s4s2 =
sqrt((current_data(current_time,4))^2-(adj_triangle_8B_s4s2)^2); % x
                    results_triangulation(current_row,1) = technologies;
                    results_triangulation(current_row,2) = current_time;
                    results_triangulation(current_row,3) = 8.2;
                    results_triangulation(current_row,4) = 11.195-
opp_triangle_8B_s4s2; % Need to subtract opp from width to obtain same x/y-axis,
then put opp value in fourth column as x
                    results_triangulation(current_row,5) = 10.40-
adj_triangle_8B_s4s2; % Need to subtract adj from length to obtain same x/y-
axis, then put adj value in fifth column as y
                    current_row = current_row +1;
                else
                end
            end
       % Remove negative values in results_triangulation and results outside of
room
```

```
       results_triangulation =
results_triangulation(results_triangulation(:,4)>0,:);
       results_triangulation =
results_triangulation(results_triangulation(:,5)>0,:);
       results_triangulation =
results_triangulation(results_triangulation(:,4)<11.195,:);
       results_triangulation =
results_triangulation(results_triangulation(:,5)<20.80,:);
       current_row = size(find(results_triangulation(:,1)>0),1)+1; % Find total
rows after removing negative values
       currentresults =
results_triangulation((results_triangulation(:,1)==technologies) &
(results_triangulation(:,2)==current_time),:);  % Save results that only contain
current time and current technology
                 if size(currentresults)>=1 % Test to see if data is zero
                     if technologies == 2 % Test to see if wifi or bluetooth
                       xy_results_triangulation_bluetooth_currentlocation =
currentresults(:,4:5); % Use x and y value in column 4 and 5
                     else
                       xy_results_triangulation_wifi_currentlocation =
currentresults(:,4:5);
                     end
                 end
         end
         current_data = Results_dynamic_bluetooth_distance; % Changing
technology and data set
     end
         % Pre-work for plotting the xy-coordinates
         wifilength = size(find(results_triangulation(:,1)==1)); % length of
results_triangulation for wifi
         results_triangulation_wifi = results_triangulation(1:wifilength,:); %
show only wifi results
         bluetoothlength = size(find(results_triangulation(:,1)==2)); % length
of results_triangulation for bluetooth
         results_triangulation_bluetooth =
results_triangulation(wifilength+1:(wifilength+bluetoothlength),:); % show only
bluetooth results

         % WiFi - If there exists multiple results for the same second,
calculate mean value of these coordinates
         wifi_unique_values = unique(results_triangulation_wifi(:,2),'stable');
% show unique values
         results_triangulation_wifi_mean = zeros(length(wifi_unique_values),6);
% allocate new variable as a table with zeros
         counter=1;
         for nr=1:length(wifi_unique_values) % for the length of unique values
```

```
             count_unique =
histc(results_triangulation_wifi(:,2),unique(wifi_unique_values)); % count the
unique values
             results_triangulation_wifi_mean(nr,1:5) =
results_triangulation_wifi(counter,1:5);
             results_triangulation_wifi_mean(nr,6) = 0; % if not merged, set
zero in column 6
             if count_unique(nr) > 1 % if there exists multiple results, then
statement is true
                 print_unique =
find(results_triangulation_wifi(:,2)==wifi_unique_values(nr));
                 start_unique = min(print_unique);
                 end_unique = max(print_unique);
                 results_triangulation_wifi_mean(nr,4) =
mean(results_triangulation_wifi((start_unique:end_unique),4)); % mean of x
values
                 results_triangulation_wifi_mean(nr,5) =
mean(results_triangulation_wifi((start_unique:end_unique),5)); % mean of y
values
                 results_triangulation_wifi_mean(nr,6) = 1; % if merged, set
value in column 6
                 counter = counter-1 + count_unique(nr); % jump to next unique
number
             end
             counter=counter+1; % increase counter by one
         end
         results_triangulation_wifi_mean =
results_triangulation_wifi_mean(results_triangulation_wifi_mean(:,6) >0,:); %
Update table to include only merged coordinates

         % Find timestamps from wi-fi data that were used in the method
         timestamps_wifi = strings(wifilength(1,1),2); % List of timestamps
         for t=1:wifilength % For the length of wifi triangulation results
relate a timestamp to the results
           timestampnumber = results_triangulation_wifi(t,2); % timestampnumber
increases when new timestamp in results_triangulation_wifi changes
           timestamps_wifi(t,1) =
Fulltable_dynamic_wifi_distance{timestampnumber,1}; % Creates a list of
timestamp related to wifi triangulation results. Here, timestamps can occur
several times
           timestamps_wifi(t,2) = results_triangulation_wifi(t,2);
         end

          % Find merged timestamps from wi-fi data that are merged
         timestamps_wifi_merged =
strings(length(results_triangulation_wifi_mean(:,1)),2); % List of timestamps
```

```
        for t=1:length(timestamps_wifi_merged(:,1)) % For the length of wifi
triangulation results relate a timestamp to the results
          timestampnumber_merged = results_triangulation_wifi_mean(t,2); %
timestampnumber increases when new timestamp in results_triangulation_wifi
changes
          timestamps_wifi_merged(t,1) =
Fulltable_dynamic_wifi_distance{timestampnumber_merged,1}; % Creates a list of
timestamp related to wifi triangulation results. Here, timestamps can occur
several times
          timestamps_wifi_merged(t,2) = results_triangulation_wifi_mean(t,2);
        end

        % Bluetooth - If there exists multiple results for the same second,
calculate mean value of these coordinates
        bluetooth_unique_values =
unique(results_triangulation_bluetooth(:,2),'stable'); % show unique values
        results_triangulation_bluetooth_mean =
zeros(length(bluetooth_unique_values),6); % allocate new variable as a table
with zeros
        counter=1;
        for nr=1:length(bluetooth_unique_values) % for the length of unique
values
            count_unique =
histc(results_triangulation_bluetooth(:,2),unique(bluetooth_unique_values)); %
count the unique values
            results_triangulation_bluetooth_mean(nr,1:5) =
results_triangulation_bluetooth(counter,1:5);
            results_triangulation_bluetooth_mean(nr,6) = 0; % if not merged,
set zero in column 6
            if count_unique(nr) > 1 % if there exists multiple results, then
statement is true
                print_unique =
find(results_triangulation_bluetooth(:,2)==bluetooth_unique_values(nr));
                start_unique = min(print_unique);
                end_unique = max(print_unique);
                results_triangulation_bluetooth_mean(nr,4) =
mean(results_triangulation_bluetooth((start_unique:end_unique),4)); % mean of x
values
                results_triangulation_bluetooth_mean(nr,5) =
mean(results_triangulation_bluetooth((start_unique:end_unique),5)); % mean of y
values
                results_triangulation_bluetooth_mean(nr,6) = 1; % if merged,
set value in column 6
                counter = counter-1 + count_unique(nr); % jump to next unique
number
            end
```

```
                counter=counter+1; % increase counter by one
            end
            results_triangulation_bluetooth_mean =
results_triangulation_bluetooth_mean(results_triangulation_bluetooth_mean(:,6)
>0,:); % Update table to include only merged coordinates

            % Find timestamps from bluetooth data that were used in the method
              timestamps_bluetooth = strings(bluetoothlength(1,1),2);
              for t=1:bluetoothlength
                  timestampnumber = results_triangulation_bluetooth(t,2);
                  timestamps_bluetooth(t,1) =
Fulltable_dynamic_bluetooth_distance{timestampnumber,1};
                  timestamps_bluetooth(t,2) = results_triangulation_bluetooth(t,2);
              end

           % Find merged timestamps from bluetooth data that are merged
            timestamps_bluetooth_merged =
strings(length(results_triangulation_bluetooth_mean(:,1)),2); % List of
timestamps
            for t=1:length(timestamps_bluetooth_merged(:,1)) % For the length of
wifi triangulation results relate a timestamp to the results
              timestampnumber_merged = results_triangulation_bluetooth_mean(t,2); %
timestampnumber increases when new timestamp in results_triangulation_wifi
changes
              timestamps_bluetooth_merged(t,1) =
Fulltable_dynamic_bluetooth_distance{timestampnumber_merged,1}; % Creates a list
of timestamp related to wifi triangulation results. Here, timestamps can occur
several times
              timestamps_bluetooth_merged(t,2) =
results_triangulation_bluetooth_mean(t,2);
            end


            % EveryXrdRow - Need to skip Wi-Fi rows in order to present the data
more nicley - Manual work

            if (length(results_triangulation_wifi_mean(:,1)) > 130 &&
length(results_triangulation_wifi_mean(:,1)) <= 150)
                          results_triangulation_wifi_mean =
results_triangulation_wifi_mean(1:15:end,:);
                          timestamps_wifi_merged = timestamps_wifi_merged(1:15:end,:);
            elseif (length(results_triangulation_wifi_mean(:,1)) > 95 &&
length(results_triangulation_wifi_mean(:,1)) <= 130)
                          results_triangulation_wifi_mean =
results_triangulation_wifi_mean(1:13:end,:);
                          timestamps_wifi_merged = timestamps_wifi_merged(1:13:end,:);
```

```
        elseif (length(results_triangulation_wifi_mean(:,1)) > 70 &&
length(results_triangulation_wifi_mean(:,1)) <= 95)
                   results_triangulation_wifi_mean =
results_triangulation_wifi_mean(1:9:end,:);
                   timestamps_wifi_merged = timestamps_wifi_merged(1:9:end,:);
         elseif (length(results_triangulation_wifi_mean(:,1)) > 50 &&
length(results_triangulation_wifi_mean(:,1)) <= 70)
                   results_triangulation_wifi_mean =
results_triangulation_wifi_mean(1:7:end,:);
                   timestamps_wifi_merged = timestamps_wifi_merged(1:7:end,:);
         elseif (length(results_triangulation_wifi_mean(:,1)) >= 28 &&
length(results_triangulation_wifi_mean(:,1)) <= 50)
                   results_triangulation_wifi_mean =
results_triangulation_wifi_mean(1:3:end,:);
                   timestamps_wifi_merged = timestamps_wifi_merged(1:3:end,:);
         elseif (length(results_triangulation_wifi_mean(:,1)) >= 18 &&
length(results_triangulation_wifi_mean(:,1)) < 30)
                   results_triangulation_wifi_mean =
results_triangulation_wifi_mean(1:2:end,:);
                   timestamps_wifi_merged = timestamps_wifi_merged(1:2:end,:);
         elseif (length(results_triangulation_wifi_mean(:,1)) > 10 &&
length(results_triangulation_wifi_mean(:,1)) < 18)
                   results_triangulation_wifi_mean =
results_triangulation_wifi_mean(1:3:end,:);
                   timestamps_wifi_merged = timestamps_wifi_merged(1:3:end,:);
         else
             % Few coordinates - Continue
         end

         % EveryXrdRow - Need to skip Bluetooth rows in order to present the
data more nicley - Manual work
         if      (length(results_triangulation_bluetooth_mean(:,1)) > 95 &&
length(results_triangulation_bluetooth_mean(:,1)) <= 150)
                 results_triangulation_bluetooth_mean =
results_triangulation_bluetooth_mean(1:13:end,:);
                 timestamps_bluetooth_merged =
timestamps_bluetooth_merged(1:13:end,:);
         elseif (length(results_triangulation_bluetooth_mean(:,1)) > 60 &&
length(results_triangulation_bluetooth_mean(:,1)) <= 95)
                 results_triangulation_bluetooth_mean =
results_triangulation_bluetooth_mean(1:9:end,:); % Every other 4
                 timestamps_bluetooth_merged =
timestamps_bluetooth_merged(1:9:end,:);
         elseif (length(results_triangulation_bluetooth_mean(:,1)) >= 27 &&
length(results_triangulation_bluetooth_mean(:,1)) <= 60)
```

```
                 results_triangulation_bluetooth_mean =
results_triangulation_bluetooth_mean(1:3:end,:); % Every other 3
                 timestamps_bluetooth_merged =
timestamps_bluetooth_merged(1:3:end,:);
         elseif (length(results_triangulation_bluetooth_mean(:,1)) >= 18 &&
length(results_triangulation_bluetooth_mean(:,1)) < 30)
                 results_triangulation_bluetooth_mean =
results_triangulation_bluetooth_mean(1:2:end,:); % Every other 2
                 timestamps_bluetooth_merged =
timestamps_bluetooth_merged(1:2:end,:);
         elseif (length(results_triangulation_bluetooth_mean(:,1)) > 10 &&
length(results_triangulation_bluetooth_mean(:,1)) < 18)
                 results_triangulation_bluetooth_mean =
results_triangulation_bluetooth_mean(1:2:end,:); % Every other 2
                 timestamps_bluetooth_merged =
timestamps_bluetooth_merged(1:2:end,:);
         else
             % Few coordinates - Continue
         end

         % Save results into six bigger tables (device specific) for plotting
         Triangulation_timestamp_devices_wifi(:,device) = "";  % Clear column

Triangulation_timestamp_devices_wifi(1:(length(timestamps_wifi_merged(:,1))),dev
ice) = timestamps_wifi_merged(:,1); % Fill table with timestamp from device
         fname =
sprintf('Output/@Plotting/Triangulation_plot_wifi_timestamp_exp%s.mat',
experiment); % Declare a variable name associated to current experiment
         save (fname, 'Triangulation_timestamp_devices_wifi'); % save variable
as mat-file

         %Triangulation_x_devices_wifi = zeros(100,10);
         Triangulation_x_devices_wifi(:,device) = 0;

Triangulation_x_devices_wifi(1:(length(results_triangulation_wifi_mean(:,1))),de
vice) = results_triangulation_wifi_mean(:,4);
         fname = sprintf('Output/@Plotting/Triangulation_plot_wifi_x_exp%s.mat',
experiment);
         save (fname, 'Triangulation_x_devices_wifi');

         %Triangulation_y_devices_wifi = zeros(100,10);
         Triangulation_y_devices_wifi(:,device) = 0;

Triangulation_y_devices_wifi(1:(length(results_triangulation_wifi_mean(:,1))),de
vice) = results_triangulation_wifi_mean(:,5);
```

```
         fname = sprintf('Output/@Plotting/Triangulation_plot_wifi_y_exp%s.mat',
experiment);
         save (fname, 'Triangulation_y_devices_wifi');

         Triangulation_timestamp_devices_bluetooth(:,device) = "";

Triangulation_timestamp_devices_bluetooth(1:(length(timestamps_bluetooth_merged(
:,1))),device) = timestamps_bluetooth_merged(:,1);
         fname =
sprintf('Output/@Plotting/Triangulation_plot_bluetooth_timestamp_exp%s.mat',
experiment);
         save (fname, 'Triangulation_timestamp_devices_bluetooth');

         %Triangulation_x_devices_bluetooth = zeros(100,10);
         Triangulation_x_devices_bluetooth(:,device) = 0;

Triangulation_x_devices_bluetooth(1:(length(results_triangulation_bluetooth_mean
(:,1))),device) = results_triangulation_bluetooth_mean(:,4);
         fname =
sprintf('Output/@Plotting/Triangulation_plot_bluetooth_x_exp%s.mat',
experiment);
         save (fname, 'Triangulation_x_devices_bluetooth');

         %Triangulation_y_devices_bluetooth = zeros(100,10);
         Triangulation_y_devices_bluetooth(:,device) = 0;

Triangulation_y_devices_bluetooth(1:(length(results_triangulation_bluetooth_mean
(:,1))),device) = results_triangulation_bluetooth_mean(:,5);
         fname =
sprintf('Output/@Plotting/Triangulation_plot_bluetooth_y_exp%s.mat',
experiment);
         save (fname, 'Triangulation_y_devices_bluetooth');


         % Plotting in seperate script

 elseif method == 3
```

## Trilateration

```
     xy_results = zeros(700,4);
     xy_results1 = zeros(700,4);
     xy_results2 = zeros(700,4);
     xy_results3 = zeros(700,4);
     xy_results4 = zeros(700,4);
     xy_results_trilateration_bluetooth_results(1,1:4) = zeros;
     xy_results_trilateration_wifi_results(1,1:4) = zeros;
     Data=Results_dynamic_wifi_distance;

     counter = 1;
     for technologies = 1:2 % loop through both Wi-Fi and Bluetooth data
           length_of_data = 1;
         for length_of_data = 1:size(Data,1)

         % Stores values from sensor 1,2,5,6 based on current row into r1..r6
          r1 = (Data(length_of_data,1));
          r2 = (Data(length_of_data,2));
          r5 = (Data(length_of_data,5));
          r6 = (Data(length_of_data,6));

 % Step 1-5 explained
     % Step 1 - The four equations for the three circles around the sensors are
as follows:
         % r1_p2 = (x-xy(1,1))^2 + (y-xy(1,2))^2; % Sensor1 radius to current
devcie
         % r2_p2 = (x-xy(2,1))^2 + (y-xy(2,2))^2; % Sensor2 radius to current
devcie
         % r5_p2 = (x-xy(5,1))^2 + (y-xy(5,5))^2;% Sensor5 radius to current
devcie
         % r6_p2 = (x-xy(6,1))^2 + (y-xy(6,6))^2;% Sensor6 radius to current
devcie
             % r1_p2 : Is the same as r1 power to 2 (r1^2)
             % format: ri^2 = (x-x1)^2+(y-y1)^2
             % xi and yi corresponds to the location of sensor i.
             % x and y is the common location of the current device in
"Results_wifi_signalstrength_double"

     % Step 2 - By expanding out the squares in each of these four equations:
         % r1_p2 = x^2 - 2*xy(1,1)*x + xy(1,1)^2 + y^2 - 2*xy(1,2)*y +
xy(1,2)^2;
         % r2_p2 = x^2 - 2*xy(2,1)*x + xy(2,1)^2 + y^2 - 2*xy(2,2)*y +
xy(2,2)^2;
         % r5_p2 = x^2 - 2*xy(5,1)*x + xy(5,1)^2 + y^2 - 2*xy(5,2)*y +
xy(5,2)^2;
         % r6_p2 = x^2 - 2*xy(6,1)*x + xy(6,1)^2 + y^2 - 2*xy(6,2)*y +
xy(6,2)^2;
             % format: ri^2 = x^2 - 2*xi*x + xi^2 + y^2 - 2*yi*y + yi^2, where i
is the nth sensor.

     % Step 3 - Subtracting the second equation from the first, the fifth
equation from the second, the sixth from the fifth and the first from the sixth:
```

```
         % 2x*(-xy(1,1)+xy(1,2)) + 2y*(-xy(2,1)+xy(2,2)) == r1^2 - r2^2 -
xy(1,1)^2 + xy(1,2)^2 - xy(2,1)^2 + xy(2,2)^2;
         % 2x*(-xy(1,2)+xy(1,3)) + 2y*(-xy(2,2)+xy(2,3)) == r2^2 - r5^2 -
xy(1,2)^2 + xy(1,3)^2 - xy(2,2)^2 + xy(2,3)^2;
         % 2x*(-xy(1,3)+xy(1,4)) + 2y*(-xy(2,3)+xy(2,4)) == r5^2 - r6^2 -
xy(1,3)^2 + xy(1,4)^2 - xy(2,3)^2 + xy(2,4)^2;
         % 2x*(-xy(1,4)+xy(1,1)) + 2y*(-xy(2,4)+xy(2,1)) == r6^2 - r1^2 -
xy(1,4)^2 + xy(1,1)^2 - xy(2,4)^2 + xy(2,1)^2;
             % The reason for doing it is because we want linear expressions.

     % Step 4  - Simplify the above linear equations by letters.
         % eqn1 = A*x + B*y = C;
         % eqn2 = D*x + E*y = F;
         % eqn3 = G*x + H*y = I;
         % eqn4 = J*x + K*y = L;

     % Step 5 - Let Matlab understand the three equations

     A = 2*(-xy(1,1)+xy(1,2));
     B = 2*(-xy(2,1)+xy(2,2));
     C = (r1)^2-(r2)^2-(xy(1,1))^2+(xy(1,2))^2-(xy(2,1))^2+(xy(2,2))^2;

     D = 2*(-xy(1,2)+xy(1,3));
     E = 2*(-xy(2,2)+xy(2,3));
     F = (r2)^2-(r5)^2-(xy(1,2))^2+(xy(1,3))^2-(xy(2,2))^2+(xy(2,3))^2;

     G = 2*(-xy(1,3)+xy(1,4));
     H = 2*(-xy(2,3)+xy(2,4));
     I = (r5)^2-(r6)^2-(xy(1,3))^2+(xy(1,4))^2-(xy(2,3))^2+(xy(2,4))^2;

     J = 2*(-xy(1,4)+xy(1,1));
     K = 2*(-xy(2,4)+xy(2,1));
     L = (r6)^2-(r1)^2-(xy(1,4))^2+(xy(1,1))^2-(xy(2,4))^2+(xy(2,1))^2;

               % Test with 4 sensors - B becomes zero... (1,2,5,6)
%             x = (C/A)-(B/A)*((G*F-D*I)/(G*E-D*H));
%             y = (C/B)-(A/B)*((H*F-E*I)/(D*H-E*G));

               % Becomes the same as equ 1
%             x = (L/J)-(K/J)*((G*F-D*I)/(G*E-D*H));
%             y = (L/K)-(J/K)*((H*F-E*I)/(D*H-E*G));

               % Wrong Equation (Sensor 1,2,5,6) (AN becomes BM...)
%             x = (C*H-B*I)/(A*H-B*G);
%             y = (C*G-A*I)/(B*G-A*H);
```

```
            % 1 Equation 1 (Sensor 1,2,5)
            x_1 = abs((C*E-B*F)/(A*E-B*D));
            y_1 = abs((C*D-A*F)/(B*D-A*E));

            % 2 Equation 2 (Sensor 2,5,6)
            x_2 = abs((F*H-E*I)/(D*H-E*G));
            y_2 = abs((F*G-D*I)/(E*G-D*H));

            % 3 Equation 3 (Sensor 5,6,1)
            x_3 = abs((I*K-H*L)/(G*K-H*J));
            y_3 = abs((I*J-G*L)/(H*J-G*K));

            % 4 Equation 4 (Sensor 6-1-2)
            x_4 = abs((L*A-K*C)/(J*B-K*A));
            y_4 = abs((L*A-J*C)/(K*A-J*B));

            % Save values of x and y into tables of results based on different
equations, remove null values and values outside of the room
            if x_1 > 0 || y_1 > 0 || isnan(x_1) || isnan(y_1)
                if x_1 > 11.195 || y_1 > 20.8
                else
                    xy_results1(length_of_data,1) = x_1; % Stores x values in
table xy_results1 from equation 1
                    xy_results1(length_of_data,2) = y_1; % Stores y values in
table xy_results1 from equation 1
                    xy_results1(length_of_data,3) = 1; % Store equation number
in column 3
                    xy_results1(length_of_data,4) = length_of_data; % Store
timestampnumber in column 4
                end
            end
            if x_2 > 0 || y_2 > 0 || isnan(x_2) || isnan(y_2)
                if x_2 > 11.195 || y_2 > 20.8
                else
                    xy_results2(length_of_data,1) = x_2; % Stores x values in
table xy_results2 from equation 2
                    xy_results2(length_of_data,2) = y_2; % Stores y values in
table xy_results2 from equation 2
                    xy_results2(length_of_data,3) = 2; % Store equation number
in column 3
                    xy_results2(length_of_data,4) = length_of_data; % Store
timestampnumber in column 4
                end
            end
            if x_3 > 0 || y_3 > 0 || isnan(x_3) || isnan(y_3)
                if x_3 > 11.195 || y_3 > 20.8
```

```
                 else
                     xy_results3(length_of_data,1) = x_3; % Stores x values in 
table xy_results3 from equation 3
                     xy_results3(length_of_data,2) = y_3; % Stores y values in 
table xy_results3 from equation 3
                     xy_results3(length_of_data,3) = 3; % Store equation number 
in column 3
                     xy_results3(length_of_data,4) = length_of_data; % Store 
timestampnumber in column 4
                 end
             end
             if x_4 > 0 || y_4 > 0 || isnan(x_4) || isnan(y_4)
                 if x_4 > 11.195 || y_4 > 20.8
                 else
                     xy_results4(length_of_data,1) = x_4; % Stores x values in 
table xy_results4 from equation 4
                     xy_results4(length_of_data,2) = y_4; % Stores y values in 
table xy_results4 from equation 4
                     xy_results4(length_of_data,3) = 4; % Store equation number 
in column 3
                     xy_results4(length_of_data,4) = length_of_data; % Store 
timestampnumber in column 4
                 end
             end

                 % Process results into two tables - Wifi and Bluetooth, use 
this to plot the data
                 if technologies == 2
                   xy_results_trilateration_bluetooth_total = 
[xy_results1(length_of_data,:); xy_results2(length_of_data,:); 
xy_results3(length_of_data,:); xy_results4(length_of_data,:)]; % Combine current 
results from all equations in.. total
                   xy_results_trilateration_bluetooth_total = 
xy_results_trilateration_bluetooth_total(xy_results_trilateration_bluetooth_tota
l(:,1)>0,:); % Remove invalid data
                   size_current_data = 
length(xy_results_trilateration_bluetooth_total(:,1)); % find number of rows in 
current data, use this to increment counter
                   if xy_results_trilateration_bluetooth_total(:,1)>1 % check if 
there exists data after filtering

xy_results_trilateration_bluetooth_results(counter:counter+(size_current_data-
1),:) = xy_results_trilateration_bluetooth_total(:,:); % Save current data in 
results
                   counter = counter + size_current_data; % keeping track on the 
next row the data will be saved in
```

```
                   else
                       continue; % Passes control to the next iteration of
lengthof_data if no data
                   end
                 else
                   xy_results_trilateration_wifi_total =
[xy_results1(length_of_data,:); xy_results2(length_of_data,:);
xy_results3(length_of_data,:); xy_results4(length_of_data,:)]; % Combine current
results from all equations in.. total
                   xy_results_trilateration_wifi_total =
xy_results_trilateration_wifi_total(xy_results_trilateration_wifi_total(:,1)>0,:
); % Remove invalid data
                   size_current_data =
length(xy_results_trilateration_wifi_total(:,1)); % find number of rows in
current data, use this to increment counter
                   if xy_results_trilateration_wifi_total(:,1)>1 % check if
there exists data after filtering

xy_results_trilateration_wifi_results(counter:counter+(size_current_data-1),:) =
xy_results_trilateration_wifi_total(:,:); % Save current data in results
                   counter = counter + size_current_data; % keeping track on the
next row the data will be saved in
                   else
                       continue; % Passes control to the next iteration of
lengthof_data if no data
                   end
                 end
         end
             length_of_data = length_of_data + 1; % for each row of data,
increment the row at the end of the loop
                 % Keep track on technology
                 if technologies == 2
                 else
                     Data=Results_dynamic_bluetooth_distance; % Change
technology
                     counter=1; % reset counter
                 end
     end

     % Pre-work for plotting
         wifilength = size(xy_results_trilateration_wifi_results(:,1)); % length
of results_triangulation for wifi
         bluetoothlength =
size(xy_results_trilateration_bluetooth_results(:,1)); % length of
results_triangulation for bluetooth
```

```
     % WiFi - If there exists multiple results within the same second, calculate
mean value of these coordinates
         wifi_unique_values =
unique(xy_results_trilateration_wifi_results(:,4),'stable'); % show unique
values
         xy_results_trilateration_wifi_merged =
zeros(length(wifi_unique_values),5); % allocate new variable as a table with
zeros
         counter=1;
         for nr=1:length(wifi_unique_values) % for the length of unique values
             count_unique =
histc(xy_results_trilateration_wifi_results(:,4),unique(wifi_unique_values)); %
count the unique values
             xy_results_trilateration_wifi_merged(nr,1:4) =
xy_results_trilateration_wifi_results(counter,1:4);
             xy_results_trilateration_wifi_merged(nr,5) = 0; % if not merged,
set zero in column 5
             if count_unique(nr) > 1 % if there exists multiple results, then
statement is true
                 print_unique =
find(xy_results_trilateration_wifi_results(:,4)==wifi_unique_values(nr));
                 start_unique = min(print_unique);
                 end_unique = max(print_unique);
                 xy_results_trilateration_wifi_merged(nr,1) =
mean(xy_results_trilateration_wifi_results((start_unique:end_unique),1)); % mean
of x values
                 xy_results_trilateration_wifi_merged(nr,2) =
mean(xy_results_trilateration_wifi_results((start_unique:end_unique),2)); % mean
of y values
                 xy_results_trilateration_wifi_merged(nr,5) = 1; % if merged,
set value in column 5
                 counter = counter-1 + count_unique(nr); % jump to next unique
number
             else
                 xy_results_trilateration_wifi_merged(nr,1) =
xy_results_trilateration_wifi_results(nr,1); % mean of x values
                 xy_results_trilateration_wifi_merged(nr,2) =
xy_results_trilateration_wifi_results(nr,2); % mean of y values
                 xy_results_trilateration_wifi_merged(nr,5) = 0; % if not
merged, set value in column 5
             end
             counter=counter+1; % increase counter by one
         end
         xy_results_trilateration_wifi_merged =
xy_results_trilateration_wifi_merged(xy_results_trilateration_wifi_merged(:,5)
>0,:); % Update table to include only merged coordinates
```

```
         % Find timestamps from wi-fi data that were used in the method
         timestamps_wifi = strings(wifilength(1,1),4); % List of timestamps
         for t=1:wifilength % For the length of wifi trilateration results
relate a timestamp to the results
           timestampnumber = xy_results_trilateration_wifi_results(t,4); %
timestampnumber increases when new timestamp in
xy_results_trilateration_wifi_results changes
           timestamps_wifi(t,1) =
Fulltable_dynamic_wifi_distance{timestampnumber,1}; % Creates a list of
timestamp related to wifi trilateration results. Here, timestamps can occur
several times
           timestamps_wifi(t,2) = xy_results_trilateration_wifi_results(t,4);
         end
          timestamps_wifi_unique = unique(timestamps_wifi,'rows'); % Find uniqe
list of timestamps - Will be used to plot timestamp related to merged data
          % Find merged timestamps from wi-fi data that are merged
         timestamps_trilateration_wifi_merged =
strings(length(xy_results_trilateration_wifi_merged(:,1)),2); % List of
timestamps related to merged data
         for t=1:length(timestamps_trilateration_wifi_merged(:,1)) % For the
length of wifi trilateration results relate a timestamp to the results
           timestampnumber_merged = xy_results_trilateration_wifi_merged(t,4); %
timestampnumber increases when new timestamp in
xy_results_trilateration_wifi_results changes
           timestamps_trilateration_wifi_merged(t,1) =
Fulltable_dynamic_wifi_distance{timestampnumber_merged,1}; % Creates a list of
timestamp related to wifi trilateration results. Here, timestamps can occur
several times
           timestamps_trilateration_wifi_merged(t,2) =
xy_results_trilateration_wifi_merged(t,4);
         end

     % Bluetooth - If there exists multiple results within the same second,
calculate mean value of these coordinates
         bluetooth_unique_values =
unique(xy_results_trilateration_bluetooth_results(:,4),'stable'); % show unique
values
         xy_results_trilateration_bluetooth_merged =
zeros(length(bluetooth_unique_values),5); % allocate new variable as a table
with zeros
         counter=1;
         for nr=1:length(bluetooth_unique_values) % for the length of unique
values
```

```
             count_unique =
histc(xy_results_trilateration_bluetooth_results(:,4),unique(bluetooth_unique_va
lues)); % count the unique values
             xy_results_trilateration_bluetooth_merged(nr,1:4) =
xy_results_trilateration_bluetooth_results(counter,1:4);
             xy_results_trilateration_bluetooth_merged(nr,5) = 0; % if not
merged, set zero in column 5
             if count_unique(nr) > 1 % if there exists multiple results, then
statement is true
                 print_unique =
find(xy_results_trilateration_bluetooth_results(:,4)==bluetooth_unique_values(nr
));
                 start_unique = min(print_unique);
                 end_unique = max(print_unique);
                 xy_results_trilateration_bluetooth_merged(nr,1) =
mean(xy_results_trilateration_bluetooth_results((start_unique:end_unique),1)); %
mean of x values
                 xy_results_trilateration_bluetooth_merged(nr,2) =
mean(xy_results_trilateration_bluetooth_results((start_unique:end_unique),2)); %
mean of y values
                 xy_results_trilateration_bluetooth_merged(nr,5) = 1; % if
merged, set value in column 5
                 counter = counter-1 + count_unique(nr); % jump to next unique
number
             else
                 xy_results_trilateration_bluetooth_merged(nr,1) =
xy_results_trilateration_bluetooth_results(nr,1); % mean of x values
                 xy_results_trilateration_bluetooth_merged(nr,2) =
xy_results_trilateration_bluetooth_results(nr,2); % mean of y values
                 xy_results_trilateration_bluetooth_merged(nr,5) = 0; % if not
merged, set value in column 5
             end
             counter=counter+1; % increase counter by one
         end
         xy_results_trilateration_bluetooth_merged =
xy_results_trilateration_bluetooth_merged(xy_results_trilateration_bluetooth_mer
ged(:,5) >0,:); % Update table to include only merged coordinates

         % Find timestamps from bluetooth data that were used in the method
         timestamps_bluetooth = strings(bluetoothlength(1,1),4); % List of
timestamps
         for t=1:bluetoothlength % For the length of bluetooth trilateration
results relate a timestamp to the results
           timestampnumber = xy_results_trilateration_bluetooth_results(t,4); %
timestampnumber increases when new timestamp in
xy_results_trilateration_bluetooth_results changes
```

```
           timestamps_bluetooth(t,1) =
Fulltable_dynamic_bluetooth_distance{timestampnumber,1}; % Creates a list of
timestamp related to wifi trilateration results. Here, timestamps can occur
several times
           timestamps_bluetooth(t,2) =
xy_results_trilateration_bluetooth_results(t,4);
         end
          timestamps_bluetooth_unique = unique(timestamps_bluetooth,'rows'); %
Find uniqe list of timestamps - Will be used to plot timestamp related to merged
data
          % Find merged timestamps from bluetooth data that are merged
         timestamps_trilateration_bluetooth_merged =
strings(length(xy_results_trilateration_bluetooth_merged(:,1)),2); % List of
timestamps related to merged data
         for t=1:length(timestamps_trilateration_bluetooth_merged(:,1)) % For
the length of wifi trilateration results relate a timestamp to the results
           timestampnumber_merged =
xy_results_trilateration_bluetooth_merged(t,4); % timestampnumber increases when
new timestamp in xy_results_trilateration_wifi_results changes
           timestamps_trilateration_bluetooth_merged(t,1) =
Fulltable_dynamic_bluetooth_distance{timestampnumber_merged,1}; % Creates a list
of timestamp related to wifi trilateration results. Here, timestamps can occur
several times
           timestamps_trilateration_bluetooth_merged(t,2) =
xy_results_trilateration_bluetooth_merged(t,4);
         end

         % EveryXrdRow - Need to skip Wi-Fi rows in order to present the data
more nicley - Manual work
         if      (length(xy_results_trilateration_wifi_merged(:,1)) > 90 &&
length(xy_results_trilateration_wifi_merged(:,1)) <= 150)
                 xy_results_trilateration_wifi_merged =
xy_results_trilateration_wifi_merged(1:13:end,:);
                 timestamps_trilateration_wifi_merged =
timestamps_trilateration_wifi_merged(1:13:end,:);
         elseif (length(xy_results_trilateration_wifi_merged(:,1)) > 60 &&
length(xy_results_trilateration_wifi_merged(:,1)) <= 90)
                 xy_results_trilateration_wifi_merged =
xy_results_trilateration_wifi_merged(1:7:end,:); % Every other 3
                 timestamps_trilateration_wifi_merged =
timestamps_trilateration_wifi_merged(1:7:end,:);
         elseif (length(xy_results_trilateration_wifi_merged(:,1)) >= 30 &&
length(xy_results_trilateration_wifi_merged(:,1)) <= 60)
                   xy_results_trilateration_wifi_merged =
xy_results_trilateration_wifi_merged(1:3:end,:); % Every other 3
```

```
                   timestamps_trilateration_wifi_merged =
timestamps_trilateration_wifi_merged(1:3:end,:);
         elseif (length(xy_results_trilateration_wifi_merged(:,1)) >= 18 &&
length(xy_results_trilateration_wifi_merged(:,1)) < 30)
                   xy_results_trilateration_wifi_merged =
xy_results_trilateration_wifi_merged(1:2:end,:); % Every other 2
                   timestamps_trilateration_wifi_merged =
timestamps_trilateration_wifi_merged(1:2:end,:);
         elseif (length(xy_results_trilateration_wifi_merged(:,1)) > 10 &&
length(xy_results_trilateration_wifi_merged(:,1)) < 18)
                   xy_results_trilateration_wifi_merged =
xy_results_trilateration_wifi_merged(1:2:end,:); % Every other 2
                   timestamps_trilateration_wifi_merged =
timestamps_trilateration_wifi_merged(1:2:end,:);
         else
             % Few coordinates - Continue
         end

         % EveryXrdRow - Need to skip Bluetooth rows in order to present the
data more nicley - Manual work
         if      (length(xy_results_trilateration_bluetooth_merged(:,1)) > 90 &&
length(xy_results_trilateration_bluetooth_merged(:,1)) <= 150)
                 xy_results_trilateration_bluetooth_merged =
xy_results_trilateration_bluetooth_merged(1:13:end,:);
                 timestamps_trilateration_bluetooth_merged =
timestamps_trilateration_bluetooth_merged(1:13:end,:);
         elseif (length(xy_results_trilateration_bluetooth_merged(:,1)) > 60 &&
length(xy_results_trilateration_bluetooth_merged(:,1)) <= 90)
                 xy_results_trilateration_bluetooth_merged =
xy_results_trilateration_bluetooth_merged(1:9:end,:);
                 timestamps_trilateration_bluetooth_merged =
timestamps_trilateration_bluetooth_merged(1:9:end,:);
         elseif (length(xy_results_trilateration_bluetooth_merged(:,1)) >= 30 &&
length(xy_results_trilateration_bluetooth_merged(:,1)) <= 60)
                 xy_results_trilateration_bluetooth_merged =
xy_results_trilateration_bluetooth_merged(1:4:end,:);
                 timestamps_trilateration_bluetooth_merged =
timestamps_trilateration_bluetooth_merged(1:4:end,:);
         elseif (length(xy_results_trilateration_bluetooth_merged(:,1)) >= 18 &&
length(xy_results_trilateration_bluetooth_merged(:,1)) < 30)
                 xy_results_trilateration_bluetooth_merged =
xy_results_trilateration_bluetooth_merged(1:3:end,:);
                 timestamps_trilateration_bluetooth_merged =
timestamps_trilateration_bluetooth_merged(1:3:end,:);
         elseif (length(xy_results_trilateration_bluetooth_merged(:,1)) > 21 &&
length(xy_results_trilateration_bluetooth_merged(:,1)) < 18)
```

```
                 xy_results_trilateration_bluetooth_merged =
xy_results_trilateration_bluetooth_merged(1:1:end,:);
                 timestamps_trilateration_bluetooth_merged =
timestamps_trilateration_bluetooth_merged(1:1:end,:);
         else
             % Few coordinates - Continue
         end

         % Save results into six bigger tables (device specific) for plotting
         Trilateration_timestamp_devices_wifi(:,device) = "";  % Clear column

Trilateration_timestamp_devices_wifi(1:(length(timestamps_trilateration_wifi_mer
ged(:,1))),device) = timestamps_trilateration_wifi_merged(:,1); % Fill table
with timestamp from device
         fname =
sprintf('Output/@Plotting/Trilateration_plot_wifi_timestamp_exp%s.mat',
experiment); % Declare a variable name associated to current experiment
         save (fname, 'Trilateration_timestamp_devices_wifi'); % save variable
as mat-file

         Trilateration_x_devices_wifi(:,device) = 0;

Trilateration_x_devices_wifi(1:(length(xy_results_trilateration_wifi_merged(:,1)
)),device) = xy_results_trilateration_wifi_merged(:,1);
         fname = sprintf('Output/@Plotting/Trilateration_plot_wifi_x_exp%s.mat',
experiment);
         save (fname, 'Trilateration_x_devices_wifi');

         Trilateration_y_devices_wifi(:,device) = 0;

Trilateration_y_devices_wifi(1:(length(xy_results_trilateration_wifi_merged(:,1)
)),device) = xy_results_trilateration_wifi_merged(:,2);
         fname = sprintf('Output/@Plotting/Trilateration_plot_wifi_y_exp%s.mat',
experiment);
         save (fname, 'Trilateration_y_devices_wifi');

         Trilateration_timestamp_devices_bluetooth(:,device) = "";

Trilateration_timestamp_devices_bluetooth(1:(length(timestamps_trilateration_blu
etooth_merged(:,1))),device) = timestamps_trilateration_bluetooth_merged(:,1);
         fname =
sprintf('Output/@Plotting/Trilateration_plot_bluetooth_timestamp_exp%s.mat',
experiment);
         save (fname, 'Trilateration_timestamp_devices_bluetooth');

         Trilateration_x_devices_bluetooth(:,device) = 0;
Trilateration_x_devices_bluetooth(1:(length(xy_results_trilateration_bluetooth_m
erged(:,1))),device) = xy_results_trilateration_bluetooth_merged(:,1);
         fname =
sprintf('Output/@Plotting/Trilateration_plot_bluetooth_x_exp%s.mat',
experiment);
         save (fname, 'Trilateration_x_devices_bluetooth');

         Trilateration_y_devices_bluetooth(:,device) = 0;

Trilateration_y_devices_bluetooth(1:(length(xy_results_trilateration_bluetooth_m
erged(:,1))),device) = xy_results_trilateration_bluetooth_merged(:,2);
         fname =
sprintf('Output/@Plotting/Trilateration_plot_bluetooth_y_exp%s.mat',
experiment);
         save (fname, 'Trilateration_y_devices_bluetooth');


         clear xy_results_trilateration_bluetooth_results; % Reset results for
next device
         clear xy_results_trilateration_wifi_results; % Reset results for next
device

     %     Plotting in separate script
 end
 end
```

Atle Malthe Sørenssen

Ensuring quality of covert police work with Wi-Fi and Bluetooth technology

NTNU
Kunnskap for en bedre verden