



THE UNIVERSITY
OF QUEENSLAND
A U S T R A L I A

INDOOR ENVIRONMENTS LOCALISATION USING ULTRA-WIDEBAND SIGNALS

by

Minh Quan Nguyen

School of Information Technology and Electrical Engineering
The University of Queensland

Submitted for the degree of Bachelor of Engineering (Honours) in
the division of Software Engineering

November 1, 2021

Minh Quan Nguyen

minhquan.nguyen@uqconnect.edu.au

November 1, 2021

Prof Amin Abbosh

Head of School

School of Information Technology and Electrical Engineering

The University of Queensland

St Lucia, QLD 4072

Dear Prof. Amin Abbosh,

In accordance with the requirements of the degree of Bachelor of Engineering (Honours) in the division of Software Engineering, I submit the following thesis entitled

“Indoor Environments Localisation Using Ultra-Wideband Signals”.

This project was performed under the supervision of Dr Guohun Zhu. I declare that the work submitted in this thesis is my own, except as acknowledged in the text and footnotes, and it has not been previously submitted for a degree at The University of Queensland or any other university or institution.

Yours Sincerely,

Minh Quan Nguyen

Minh Quan Nguyen

Acknowledgements

I would like to express my sincerest gratitude to Dr Guohun Zhu, who is the supervisor of this thesis project. He has provided me with guidance, comprehensive direction, and constructive advice since the initial stages of the project. With his help, I have enhanced my expertise in a variety of areas, such as signal processing and machine learning, which are critical for completing this thesis project. Also, I want to thank Raed Shubair, Nazar Ali, and Mohamed AlHajri for approving the use of the dataset for study and research purposes. In closing, I would like to extend my deepest thanks to my family and friends for all their spiritual support and encouragement throughout my university journey.

Abstract

In today's world of fast-paced world of information technology, positioning is no doubt a major topic. It is apparent that location positioning involves various daily activities such as shopping online or finding directions to a specific spot. All of these tasks are achieved thanks to the Global Positioning System (GPS) installed the digital devices, and they are collectively known as outdoor positioning. At the moment, outdoor navigation systems are already highly effective; however, there are certain challenges when it comes to indoor localisation. Being able to detect a person or an object in a closed area may benefit in a variety of ways, from finding the right store in a shopping mall or a specific product in a supermarket. There are several technologies that have been applied for indoor positioning, and some of the most familiar ones are Bluetooth, Wireless Fidelity (Wi-Fi), and surveillance camera. Each of the technologies listed above has its advantages and disadvantages, and no technology is perfect in every scenario. This thesis project focuses on exploiting a reasonably new technology that is Ultra-Wideband (UWB). UWB is a signal-based technology, and it is expected to bring new aspects into the field as it has outstanding strengths. The indoor positioning of this project is analysed on a uniform grid location of a floor of a building. The UWB signals are fed into algorithms, machine learning models or neural networks to forecast the final results. In addition, the efficiency of the proposed technique in this thesis will be compared to that of other existing methods in order to determine whether it is suitable for any particular case.

Table of Contents

Acknowledgements.....	v
Abstract	vii
List of Figures.....	x
List of Tables.....	xii
Abbreviations.....	xiii
1 Introduction.....	1
2 Background Research	4
2.1 Positioning Techniques	4
2.1.1 Received Signal Strength Indication.....	4
2.1.2 Time of Arrival.....	4
2.1.3 Time Difference of Arrival.....	6
2.1.4 Angular of Arrival	7
2.1.5 Power Profile Delay.....	7
2.2 Algorithms in Indoor Localisation.....	8
2.3 Scattering Parameter	8
2.4 Fourier Transform.....	9
3 Existing Methods and Related Works	13
3.1 Dataset Summary	13
3.2 Indoor Rooms Classification.....	15
3.3 Similar Works	17
4 Indoor Environments Classification.....	18
4.1 Data Analysis and Preparation.....	18
4.2 Data Visualisation.....	21
4.3 Machine Learning Classification Algorithms.....	23
4.4 Deep Learning Neural Networks	25
4.5 Results Comparison and Discussion.....	29

5	Grid Location Positioning Algorithm	31
5.1	Frequency Domain and Time Domain Comparison	31
5.2	Step 1 - Time Domain Conversion	32
5.3	Step 2 – Divide into Zones.....	34
5.3.1	Step 2a – Divide into Zones based on TOA and RSS.....	34
5.3.2	Step 2b – Divide into Zones based on AOA	36
5.4	Step 3 – Zones Classification.....	37
5.5	Step 4 – Combine Zones	40
5.6	Results and Discussion.....	42
5.7	Compared with Existing Methods.....	46
6	Conclusion and Future Work	48
6.1	Conclusion	48
6.2	Future Work	49
7	Appendix	50
7.1	Exemption Approval.....	50
7.2	Dataset and Code	51
7.3	Results	51
8	Bibliography	55

List of Figures

Figure 2.1 Simple TOA system setup with three base stations in a two-dimensional case [9]	5
Figure 2.2 Location positioning system with TDOA [10]	6
Figure 2.3 AoA-based system with two base stations [12]	7
Figure 2.4 An example of a 2-port network and S-parameter matrix	9
Figure 2.5 Converting to the time domain using Hermitian symmetry [17]	11
Figure 3.1 Floorplan of the area where the dataset is conducted	14
Figure 3.2 Grid area with each location's naming	15
Figure 3.3 $C^{(n)}$ of CTF [18]	16
Figure 3.4 $C^{(n)}$ of FCF [18]	16
Figure 3.5 Flowchart of the algorithm proposed in [21]	16
Figure 3.6 (a) Stage 1 (without Karman Filter) (b) Case 2 (with Karman Filter) [22]	17
Figure 4.1 (a) Magnitude (dB) of S_{11} (b) Magnitude (dB) of S_{21} at location (8, 11) of four rooms	18
Figure 4.2 (a) Real part of S_{11} (b) Imaginary part of S_{11} (c) Real part of S_{21} (d) Imaginary part of S_{21} at the location (8, 11) in four rooms	19
Figure 4.3 (a) Average magnitude (dB) of S_{11} (b) Average magnitude (dB) of S_{11} of 196 locations (signals from lab room)	20
Figure 4.4 Observations projected onto three PCA components	22
Figure 4.5 UMAP projection of the dataset	23
Figure 4.6 Error rate vs value of k in KNN	25
Figure 4.7 Comparison between RNN and LSTM	26
Figure 4.8 Flowchart of CNN-LSTM	27
Figure 4.9 Architecture of a block of ConvLSTM	27
Figure 4.10 Learning/Loss Curves (a) CNN-LSTM (b) ConvLSTM	29
Figure 5.1 Two signals with different frequencies in the frequency domain and time domain [32]	31
Figure 5.2 Converted time-domain signals (a) From 0 to 5988 ns (b) From 0 to 100 ns	33
Figure 5.3 Four combinations of zones after applying Step 2a on the signals from four rooms (a) Laboratory (b) Corridor (c) Lobby (d) Sports Hall	35
Figure 5.4 Three combinations of zones after applying Step 2b on the signals from three rooms (a) Laboratory (b) Lobby (c) Corridor	37
Figure 5.5 Flowchart of Step 3	38

Figure 5.6 The labels of the location (8, 3) after Step 3 (a) Laboratory's zones combination from Step 2a (b) Corridor's zones combination from Step 2a (c) Lobby's zones combination from Step 2a (d) Sports hall's zones combination from Step 2a (e) Laboratory's zones combination from Step 2b (f) Lobby's zones combination from Step 2b (g) Laboratory's zones combination from Step 2b	39
Figure 5.7 Flowchart of the algorithm of Step 4	41
Figure 5.8 The prediction of the algorithm for each location in the test set (1 st fold)	43
Figure 5.9 Example of a prediction that the size is too large	44
Figure 7.1 Exemption approval.....	50
Figure 7.2 The prediction of the algorithm for each location in the test (2 nd fold)	51
Figure 7.3 The prediction of the algorithm for each location in the test set (3 rd fold)	52
Figure 7.4 The prediction of the algorithm for each location in the test set (4 th fold)	53
Figure 7.5 The prediction of the algorithm for each location in the test set (5 th fold)	54

List of Tables

Table 1.1 Different technologies of indoor localisation system [2]	2
Table 4.1 Example of the two-dimensional array of each data point	21
Table 4.2 Explained variance of three PCA components	22
Table 4.3 The configurations and accuracy of three classification models.....	24
Table 4.4 Accuracy of CNN-LSTM and ConvLSTM.....	29
Table 4.5 Comparison between S_{11} and S_{21}	30
Table 5.1 Description of each zone in Step 2b	36
Table 5.2 Results of Zones Classification (Step 3)	42
Table 5.3 Three criteria for evaluating the performance of the grid positioning algorithm	43
Table 5.4 Comparison of two strategies	45

Abbreviations

Term	Definition
GPS	Global Positioning System
BLE	Bluetooth Low Energy
Wi-Fi	Wireless Fidelity
RFID	Radio Frequency Identification
UWB	Ultra-Wideband
RSS	Received Signal Strength
RSSI	Received Signal Strength Indication
TOA	Time of Arrival
NLOS	Non-line-of-sight
LOS	Line-of-sight
TDOA	Time Difference of Arrival
AOA	Angular of Arrival
S-Parameter	Scatter Parameter
FT	Fourier Transform
IFT	Inverse Fourier Transform
DCF	Discrete Fourier Transform
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
CIR	Channel Impulse Response
CTF	Channel Transfer Function

DC	Direct Current
VNA	Vector Network Analyser
PCA	Principal Component Analysis
UMAP	Uniform Manifold Approximation and Projection
t-SNE	t-Distributed Stochastic Neighbour Embedding
KNN	K-Nearest Neighbours
SVM	Support Vector Machine
RFC	Random Forest Classifier
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
CNN-LSTM	Convolutional Neural Network with LSTM
ConvLSTM	LSTM with Convolutional Operation

Chapter 1

1 Introduction

It cannot be denied that the area of location positioning has grown in popularity in recent years as a result of its favourable influences on daily situations. Basically, this field can be divided into two main sub-fields: outdoor positioning and indoor positioning. Outdoor positioning is a familiar technology with various applications, and the most popular one is to detect the position of humans. Global System Positioning (GPS) is no doubt the most popular system, and it is installed in almost every digital gadget (such as smartphones or laptops). It is relatively simple to identify the user's location using GPS and utilise that information to serve them again in various situations (some software applications such as food delivery and route finder need users' position to operate). On the other hand, indoor positioning is less well-known due to the difficulties of perfecting it. GPS and other satellite-based technologies are powerful and precise in open spaces, yet they show limitations in indoor environments. As stated in [1], GPS's performance remarkably decreases when it is tested in indoor areas (namely multi-floor buildings or underground garages) because of signal blockage from room objects. Detecting people on different floors of a building, as an example, is another task that is challenging for GPS to achieve. Therefore, indoor positioning systems are developed to handle these circumstances where GPS's accuracy is poor.

An indoor positioning system is made up of devices and technologies used to determine people or objects' position in a closed environment. There are many situations where an indoor tracking system has been applied:

- Locating vehicles in a crowded underground parking lot.
- Finding a particular store in a plaza.
- Managing kids in schools to ensure their safety.
- It is used to learn how workers and forklifts roam throughout a warehouse. The logistics management server receives the location data for analysis. Hence, it is simpler to optimise areas of the operation, such as equipment placement and worker/forklift positioning, resulting in a more efficient working environment.
- Finding out who is in the same room at the same time (Coronavirus-19 is a prominent topic during the time of this thesis project).

CHAPTER 1 INTRODUCTION

There are various technologies that are used for implementing an indoor positioning system, and they can be categorised into four groups: radio-based, optical, magnetic, and acoustic (**Table 1.1**). It is essential to state that each technology listed below has its own set of advantages and disadvantages and may be used in various situations.

Table 1.1 Different technologies of indoor localisation system [2]

Category	Technology
Radio-based	Bluetooth
	Wireless Fidelity (Wi-Fi)
	Radio Frequency Identification (RFID)
	Ultra-Wideband (UWB)
Optical	Surveillance Cameras
	Light Detection and Ranging (LiDAR)
	Visible Light Communications (VLC)
	Infrared (Electromagnetic Waves)
Acoustic	Ultrasound
Magnetic	Magnetic Strength

1. Bluetooth: Since most mobile phones currently have Bluetooth built-in, Bluetooth-based systems are extensively applied in a variety of indoor situations. These systems used beacons (or tags) to detect a position in an indoor environment. iBeacon from Apple, as an example, is a small widget that sends Bluetooth Low Energy (BLE) signals to other devices within its distance [3]. A standard system includes Bluetooth scanners, which have the responsibility to scan consecutively for Bluetooth signals generated by digital devices. The location of the device is recorded in the scanner, and with multiple scanners placed, the specific location of that device can be approximately estimated (the estimation is obtained with the signal strength and transmission time) [4]. Some characteristics of Bluetooth are simple to set up and low energy consumption. However, Bluetooth is prone to hacking because it is a widely used technology.
2. Wireless Fidelity (Wi-Fi): Regarding indoor localisation systems, the utilisation of Wi-Fi is broadly similar to Bluetooth. Wireless access points are set up in the area in order to receive the signals from the transmitters. Because Bluetooth is particularly

built for short-distance wireless connectivity, Wi-Fi shows poorer accuracy in comparison with Bluetooth (the accuracy range of Wi-Fi is from 5 m to 15 m) [1, 4]. This technology's strength is its universal adoption, as seen by its widespread usage in public areas. However, similar to Bluetooth, it is a standard technology and is highly vulnerable to cyber threats.

3. Radio Frequency Identification (RFID): Readers, tags, and the frequency protocol for communication between them make up this technology. RFID tags fall into two main categories: active and passive, and their objective is to transfer signals to the readers. In order to function reliably, the RFID system requires a large number of readers and tags (the higher the number, the higher the accuracy) [4]. As a result, the expense of deploying tags/readers and the fact that people must carry a tag along are the considerable drawbacks of RFID.
4. Optical Technologies: These technologies can be categorised into two classes: surveillance and egocentric vision [5]. The advantage of vision-based technologies is that they can visualise people and objects in the scene. Moreover, if they are integrated with machine learning models, it is able to achieve impressive features such as facial recognition. In return, the user experiences are not positive as some people have privacy concerns with surveillance cameras.

This thesis project focuses on a relatively new radio-based technology, which is Ultra-Wideband (UWB). UWB systems were originally designed as a military tool for inter-troop communication before the 21st century. However, in recent years, UWB has been deeply researched and exploited for the field of wireless communication. UWB is specified by the Federal Communications Commission as a technology that has a bandwidth rate of more than 20% or permits transmission with a bandwidth equal to or higher than 500 Hz [6]. UWB may be classified into two categories based on this definition: ultra-high data rate applications and low data rate applications (suitable for indoor positioning systems and communication purposes).

UWB stands out from other technologies because it presents the following benefits [6, 7]:

- UWB signals are immune to multipath.
- Because of the large bandwidth, UWB signals do not conflict with others.
- UWB is significantly more accurate in indoor positioning than Bluetooth and Wi-Fi (the average accuracy of UWB is roughly 10 cm).
- The power consumption of UWB is minimal.

UWB's advantages, when combined with other techniques, will be a potential option for not just indoor localisation but also other wireless applications.

Chapter 2

2 Background Research

2.1 Positioning Techniques

2.1.1 Received Signal Strength Indication

Receive Signal Strength (RSS) is the magnitude of a radio signal from the transmitter to the receiver, and it is inversely proportional to the fixed distance between two devices. Received Signal Strength Indication (RSSI) is a metric for evaluating RSS. Even though there is no uniformly established relationship between RSSI and the power intensity (RSSI is often defined by the manufacturers), it is self-evident that the higher the value of RSSI, the more powerful the signal. The main advantages of RSSI are its simplicity, low cost, and low power consumption. Nevertheless, whereas RSSI performs well in basic contexts, it performs poorly in high-cluttered environments (noise generated by obstructions and multipath are some typical elements that seriously impact the strength of signals) [7]. Consequently, due to its simplicity and lack of effectiveness, it is regularly used in combination with other algorithms and optimisation methods, depending on the indoor environment's terrain. For example, RSSI-based methods utilise the value of RSSI to calculate the distance and pass it to more advanced positioning algorithms (such as min-max algorithm, multilateration algorithm, and range-based algorithm) for position estimation [8]. Summing up, despite the fact that the performance of RSSI is not excellent, it is definitely worth considering.

2.1.2 Time of Arrival

Time of Arrival (TOA) is as basics as its name implies; it makes use of the arrival time of the signals from the transmitter to the receiver to determine the distance between them or estimate the position of the target. The distance from one transmitter to one receiver can be calculated using the signal's travel time by the following formula [9]:

Equation 2.1

$$d = \int_P ds = \int_{T_i}^{T_0} c dt = c_{average}(T_0 - T) \stackrel{c \text{ is constant}}{\cong} c(T_0 - T)$$

where:

d: distance

c: speed of light of the path

$c_{average}$: is the average value of c , and it is equal to c if c is constant

T_0 : is the time of beginning transmission
 T : is the time of reception

In a two-dimensional environment, the distance d can equally be interpreted as the radius around the receiver. Therefore, in order to obtain the precise position of the target, there need to be several receivers installed in the area. The intersection point of those receivers' radius ranges reveals the target's unique location (**Figure 2.1** visualises this approach).

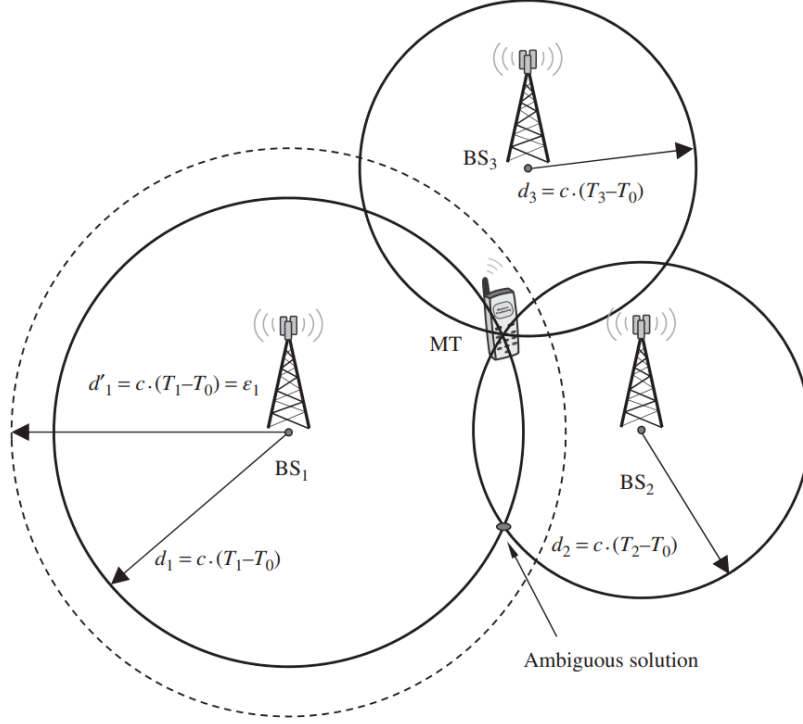


Figure 2.1 Simple TOA system setup with three base stations in a two-dimensional case [9]

As in **Figure 2.1**, the intersection of two base stations (or transmitters) provides two possible points for the target; hence the third base station is required to ensure the method's accuracy. In other words, there must be at least three base stations for this system and the more base stations, the more reliable the result. Moreover, this figure also demonstrates the weakness of the system, which is handling noise. The dashed circle of BS₁ is the radius range that takes noise error into consideration. Consequently, no point exists where three radius ranges cross. Finally, TOA systems must meet two following extra conditions in order to function correctly [10]:

- The system's devices operate properly in synchrony (this is complicated to obtain).
- Each base station is uniquely identified for further computation.
- When measuring the propagation delay for non-line-of-sight (NLOS) propagation, it is necessary to account for reflection errors (the case in **Figure 2.1** is discussed assuming the line-of-sight (LOS) propagation).

2.1.3 Time Difference of Arrival

Time Difference of Arrival (TDOA) is a different technique that is also based on the time delay associated with signal transmission. TDOA, like TOA, requires at least three base stations, but it calculates the target's position by comparing the difference between the arrival time of separate arriving signals. Assume the transmitter emits signals a and b collected by the receiver at two different time stamps, T_a and T_b . The difference in distance can be calculated by the formula below [9]:

Equation 2.2

$$\Delta d_{a,b} = d_a - d_b = c(T_a - T_0) - c(T_b - T_0) = c(T_a - T_b) = c\Delta T_{a,b}$$

Different from TOA, where the distance d forms a circle radius range around the base station, TDOA describes locations with equivalent distance differences to the base station. This definition is identical to the definition of a hyperbola in a two-dimensional space. A hyperbola is a collection of points that share the same value for the distance difference between two fixed points. Accordingly, the radius range of a base station in this technique has a shape of a hyperbola (the setup is described in **Figure 2.2**). In **Figure 2.2**, point P is the intersection of two hyperbolas from two base stations, B and C , so it is the location of the target.

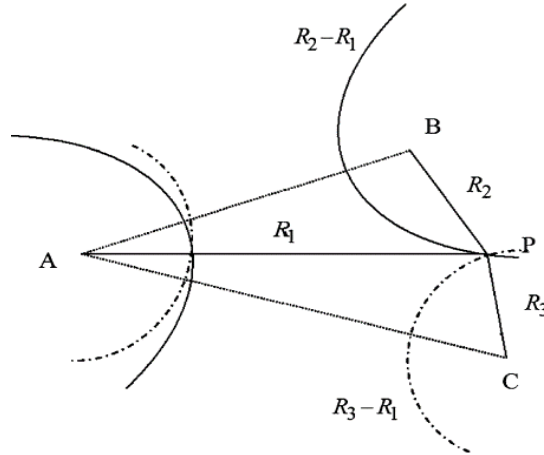


Figure 2.2 Location positioning system with TDOA [10]

As stated in [11], the primary drawback of TDOA is the significant energy consumption associated with the large number of base stations involved (this is an issue of TOA, as well). To summarise, TDOA and TOA share several characteristics, and both face some difficulties with NLOS propagation. There are two main factors of both techniques:

- In order to achieve the highest performance, flawless synchronisation is the key factor.
- While increasing the sample rate helps enhance accuracy, it might backfire if it is done excessively. This is the sole reason why the massive bandwidth of UWB is well-suited for both techniques.

2.1.4 Angular of Arrival

While TOA and TDOA make use of the propagation time delay for positioning, Angular of Arrival (AOA) uses another equally important variable, the direction of the signal. The direction of the signal is determined by measuring its angle when it arrives at the receiver. **Figure 2.3** illustrates a basic setup of an AOA-based system. In this system, there are two base stations BS_1 and BS_2 , and the distance between them is L_{12} (AOA is conditional on knowing this distance in advance). This distance, along with the two angles α_1 and α_2 of the signal sent to the two base stations, trigonometry is then used to calculate the position of the target.

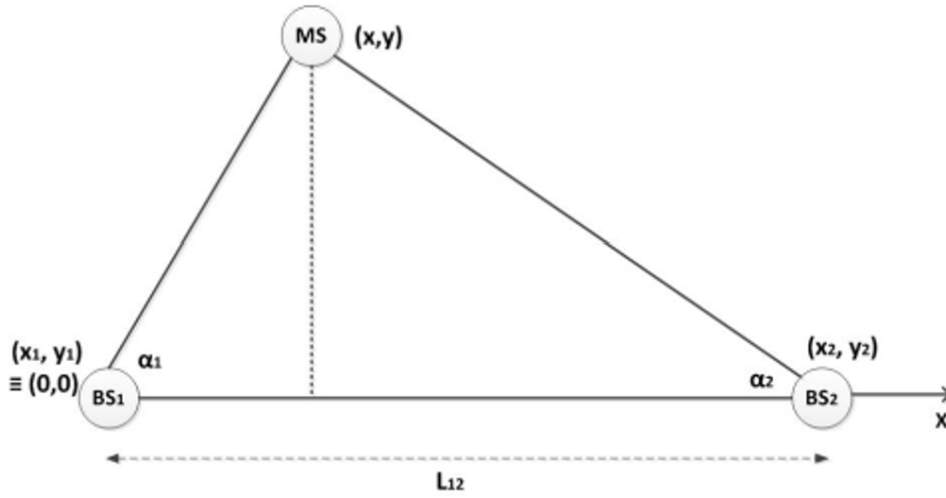


Figure 2.3 AoA-based system with two base stations [12]

Compared to ToA and TDOA, in terms of the number of base stations, AOA only requires two, while both ToA and TDOA need a minimum of three to function properly. Nonetheless, in practice, more than two base stations are placed to increase accuracy and minimise interference. Another advantage of this method is that it does not require precise synchronisation between the devices (which is a challenging factor to achieve). However, directional antennas are essential for AOA to measure the angle correctly. Finally, according to [9], although AOA can be applied in three-dimensional cases, the computation becomes more complex and requires the involvement of additional variables.

2.1.5 Power Profile Delay

An additional property of a signal that might be taken into account for location reasons is its power. The reception of multiple duplicates of the sent signal is caused by multipath propagation, and these duplicates come with varying propagation delays and signal levels. Because of uncertain proportionality constants or a lack of receiver calibration, using absolute power measurements for positioning are challenging to accomplish. This is a similar

challenge when measuring absolute propagation delays since synchronisation between devices is strictly required. It is, therefore, reasonable to use power ratios between the various arrival paths, which are relative values in this situation, to solve the problem.

2.2 Algorithms in Indoor Localisation

To perfect indoor positioning systems, the above techniques alone are insufficient; they must be integrated with other advanced algorithms and *fingerprinting* is a prevalent one. The *fingerprinting* algorithm is divided into two phases: offline and online. In the offline phase, the features of the signals acquired at many sites are stored in the database. These features are called *location fingerprints*, and they are used for training the machine learning models. There are many machine learning algorithms that are commonly applied in this situation, such as K-Nearest Neighbour and Support Vector Machine (for cohesion and coherence purposes, definitions of machine learning algorithms are covered in greater detail in later sections where they are directly applied). Subsequently, the trained machine learning models will be tested with a different fingerprint database or in a real-time scenario during the online phase.

Generally, these features collected during the offline phase include RSSI, TOA, and AOA (RSSI is widely selected since it is the easiest one to be obtained) [13]. Since the accuracy is strongly reliant on the chosen feature, two or more features are often applied to extract the maximum value from the signals. For instance, the work from [14] utilises both RSS and TOA data, with TOA being used for small areas and RSS for large areas.

2.3 Scattering Parameter

Previously, network metrics such as Y-parameters and Z-parameters most common tools to evaluate the circuit's functionality. However, at considerably high frequencies, voltage and current become increasingly difficult to link to the performance of the network and scattering parameters are suitable to tackle this issue. Scattering parameters (also called S-parameters) present the input-output connection across ports in a linear electrical system. Since S-parameters can prevent the issues when dealing with large frequencies, they are commonly utilised when working with UWB. S-parameters vary with the frequency, so for any S-parameter values given, the frequency must be indicated.

The S-parameter matrix for a two-port network is possibly the most often employed, and it can also be used to serve as the foundation for creating higher-order matrices for bigger networks (networks with more ports). A two-port network is an electrical circuit comprising

two pairs of endpoints for connecting to external networks. The example of a two-port network and its corresponding S-parameter matrix are displayed in **Figure 2.4**.

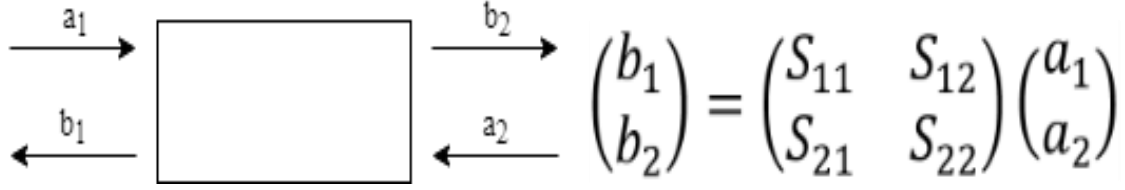


Figure 2.4 An example of a 2-port network and S-parameter matrix

The convention of the name of each S-parameter is: the first number denotes the port from which the signal originates, while the second number indicates the port from which the signal arrives. For example, S_{12} is the measurement of the signal transmitted from port 1 to port 2, while S_{11} is the reflected signal as the input port and output port are identical. The following are general explanations of each S-parameter [15]:

- S_{11} and S_{22} – Reflection (or return loss), in which S_{11} is the input return loss, and S_{22} is the output returns.
- S_{21} and S_{12} – Gain (or insertion loss), in which S_{21} presents the forward gain and S_{12} is the reverse gain.

2.4 Fourier Transform

A transform is a mapping between two sets of data and domain. In general, Fourier Transform (FT) is the formula that answers the question, “Is it possible to build a mapping function to express the frequency domain of a function f in the time domain?”. Regarding the field of signal processing, it converts information from a signal’s time domain to its frequency domain. The Inverse Fourier Transform (IFT), on the other hand, is designed to convert in the opposite way. These two domains convey the exact data/information, but in various cases, one method of representation may be preferable to the other (the comparison between the time domain and the frequency domain of a signal is further discussed in **Section 5.1**). The FT and IFT formulas are listed below:

Equation 2.3

$$FT: \hat{G}(f) = \int_{-\infty}^{\infty} G(t) e^{-2\pi i f t} dt$$

$$IFT: G(t) = \int_{-\infty}^{\infty} \hat{G}(f) e^{2\pi i f t} dt$$

$$e^{2\pi i f t} = \cos(2\pi i f t) + i \sin(2\pi i f t).$$

CHAPTER 2. BACKGROUND RESEARCH

Considering the IFT formula, the transformation from the frequency domain to the time domain can be divided into the following steps:

1. $\hat{g}(f)$ is the frequency domain function (in a complex form with both amplitude and phase) depends on frequency.
2. $e^{2\pi if t}$ is the Euler's formula that represents the sine and cosine waves.
3. With the values of frequency range from $-\infty$ to ∞ , $\hat{g}(f)$ is multiply by $e^{2\pi if t}$.
4. Finally, the sum of all waves at every frequency is the time domain. On the contrary, in the forward direction, the time domain signal is divided by $e^{2\pi if t}$, which is a combination of sine and cosine waves, results in the frequency domain (it is not a good practice to put the complex number in the numerator, that is the reason why in the FT formula above, the time domain is multiplied with $e^{-2\pi if t}$).

In the FT and IFT formulas above, the integral is run from $-\infty$ to ∞ , but in practice, the signal data is acquired in a finite timespan as points in an interval. Therefore, in order to perform FT on a discrete set of points, Discrete Fourier Transform (DFT) was established. The DFT formula is shown below, and it differs slightly from the FT formula:

Equation 2.4

$$DFT: G_k = \sum_{n=0}^{N-1} x_n * e^{-\frac{i2\pi kn}{N}}$$
$$\frac{k}{N} \triangleq F \text{ and } n \triangleq t$$

Instead of running the integral from $-\infty$ to ∞ , the DFT formula calculates the summation for each sample point. The frequency coefficients are now limited to a collection of frequency bins specified by the sampling frequency and the total number of sample counts. As this case is discrete, frequency f and time t are now described by the number of samples N , sample rate k , and sample point n . In order to calculate the DFT in a fast and efficient way, Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) algorithms were developed. FFT is regarded as one of the most important algorithms ever invented, and it has a great impact on many areas such as digital communication, audio signal, and images processing. The standard formula of DFT has a complexity $O(n^2)$, whereas FFT shows the complexity of $O(n \log(n))$, which is much faster. Accordingly, FFT is always used for calculating DFT.

If the time domain is a real function, the FT of that function must follow Hermitian symmetry (this is because when FT is applied on a real-valued function, the result obtained is a complex function) [16]. In mathematics, a Hermitian symmetry function is one that is

equivalent to its complex conjugate, but the sign of the variable is opposite. With the definition and formula, it can be inferred that a function follows Hermitian symmetry if and only if the real component of it is an even function and its imaginary part is an odd function (alternatively, the magnitude is even while the phase is odd in this situation).

Equation 2.5

$$f(-x) = f^*(x)$$

There are two popular techniques to transfer a signal to the time domain in signal analysis: one is related to Hermitian symmetry, and the other is based on complex conjugate. The first technique adheres to Hermitian symmetry by applying zero-padding out from the lowest frequency point to the direct current (DC) component. Subsequently, the conjugate complex of the zero-padded data is taken and added as a reflection in negative frequencies. As a result of this step, the signal data is now symmetric with the DC component at the middle point. Finally, the output is converted to the time domain using IFFT (the entire process is illustrated in **Figure 2.5**). The received signal has a time period that is double that of the signal obtained using the baseband method. This property enhances accuracy, and it is a considerable benefit because one of the goals of UWB channel modelling is to correctly distinguish the distinct signal pathways.

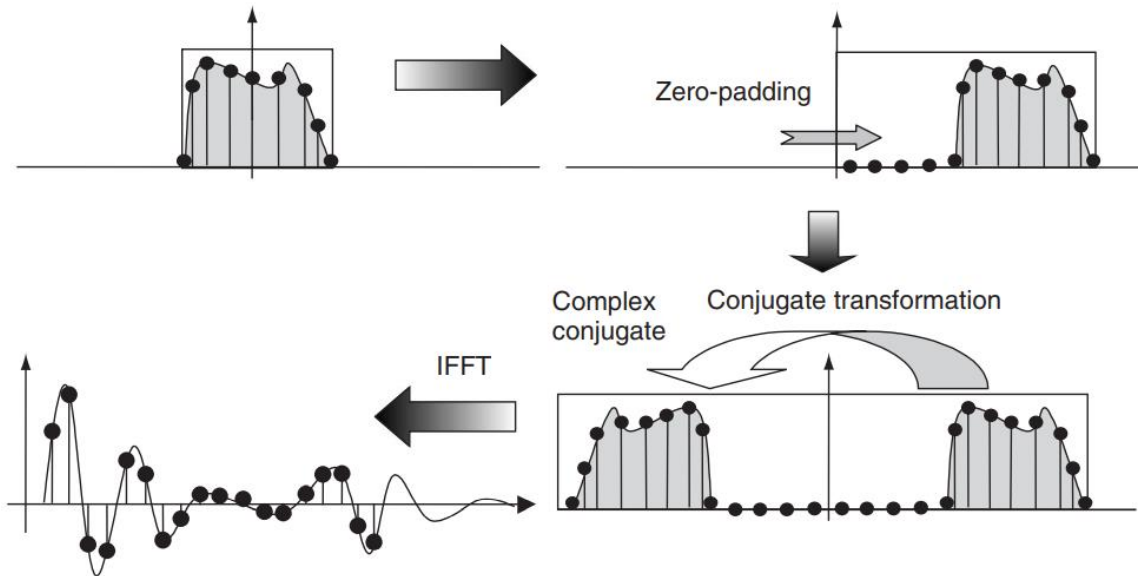


Figure 2.5 Converting to the time domain using Hermitian symmetry [17]

The steps in the second technique, which only use conjugate complex, are fairly similar to the first one. It takes the conjugate complex of the original signal and reflects it in negative frequencies. The most noticeable difference from the first method is that it does not employ zero padding. Next, just the left-hand side of the data is fed into IFFT for time-domain conversion. This technique produces results that are somewhat comparable to those reported with the first technique [17]. However, in some circumstances, the second technique is

CHAPTER 2. BACKGROUND RESEARCH

preferable thanks to its light complexity. Due to the fact that just the left-hand side of the signal is treated and there is no zero padding, the size of the output is significantly less. This property simplifies any calculations or algorithms applied to the output.

Supposed the signal data in the frequency domain, which is collected from a measuring device or via simulation, consists of N sample points evenly distributed from f_{min} to f_{max} . After that, follow the steps outlined above, zeros are added (using the same step size as the frequency sample points), and conjugate complex is taken as a reflection in negative frequencies. Lastly, IFFT is applied to produce the signal in the time domain, whose time frame has the maximum point t_{max} and the time step Δt . Given that: N is the number of sample points in the original dataset, f_{min} is the start point of frequency and f_{max} is the endpoint of frequency; the formulas of calculating those related variables are described below:

Equation 2.6

$$\text{Frequency step: } \Delta f = (f_{max} - f_{min})/N$$

$$\# \text{ of zeros added: } N_0 = (N/(f_{max} - f_{min})) * f_{min}$$

$$\# \text{ of points after reflecting complex conjugate: } N_{total} = N * 2 + N_0$$

$$\text{Maximum time point: } t_{max} = 1/\Delta f$$

$$\text{Time step: } \Delta t = 1/(\Delta f * N_{total}) = t_{max}/N_{total}$$

In some cases, when a finer resolution of the time domain is needed, a great number of N and a high value of f_{max} are necessary. On the other hand, suppose N is not large enough as desired; in that case, it can be increased multiple times by resampling using interpolation (considering the pattern of the original data is not rough and the vital information such as peaks are preserved).

Chapter 3

3 Existing Methods and Related Works

3.1 Dataset Summary

Working with the dataset provided in [18, 19] is the subject of this thesis project. The experiment was conducted on a particular floor in the building at Khalifa University campus in Sharjah, United Arab Emirates. The tests were carried out to study the Wi-Fi bands, primarily channels 1, 6, and 11 of the IEEE 802.11g-2003 specifications. The main device of the experiment is the ZVB14 vector network analyser (VNA) from Rhode and Schwartz, and it was used to measure the frequency domain of the signals. Along with that VNA, a computer script was created to measure ten sweeps in a series automatically. Each sweep has 601 frequency sample points in total, covering a bandwidth of 100 MHz (from 2.4 GHz to 2.5 GHz, and the frequency step size is $100 / 601 = 0.167$ MHz). The devices that make up the complete system are as follows:

- A VNA for evaluating the frequency domain of the signals.
- Low-loss RF cables.
- Transmitters and receivers with omnidirectional antennas (all transmitters and receivers are 1.5 meters in height).

The entire floor, where the experiment was conducted, is planned out in a 14*14 uniform grid, which means that there are 196 grid sites in total. In the original dataset, the naming convention for grid location is described in the following: ‘Location 0205’, for example, is the 3rd cell from the top (starts at 0) and the 6th from the left (starts at 0). To put it another way, the name of the grid location contains two parts, the first part is the *y-coordinate* (but it starts from the top), and the second part is the *x-coordinate* (counting from the left as usual). This naming scheme differs from the usual, which can be confusing for readers and inconvenient for the programming process. As a result, the naming convention for each grid position will be adjusted as follows: the first part is the *x-coordinate* (starts at 0 from left to right), while the *y-coordinate* is the second part (starts at 0 from bottom to top). The floor plan blueprint, arrangement of transmitters and receivers is described in **Figure 3.1**, and the numbering flow is depicted in **Figure 3.2**. Signals from the 196 grid locations were measured in four separate rooms on that floor: the lobby, and the sports hall (both are low cluttered), the corridor (moderate cluttered), and the laboratory (extremely cluttered).

CHAPTER 3. EXISTING METHODS AND RELATED WORKS

Because this VNA is a 2-port network, there are four individual S-parameters: S_{11} , S_{12} , S_{21} , and S_{22} (the detailed discussion of S-parameters can be found in Section 2.3). In this experiment, all channel impairments were minimised as it was carried out during a period of reduced activity (there was no significant movement in the surroundings of the transmitters and receivers.). As a result, S_{12} and S_{21} share the same value (this behaviour is called “Channel Symmetry”, and it has already been proved in [20]). Since “Channel Symmetry” only occurs under ideal circumstances, channels impairments must always be addressed in practice. To conclude, the dataset contains the following five parameters:

- Real component of S_{11} .
- Imaginary component of S_{11} ,
- Real component of S_{21} .
- Imaginary component of S_{21}
- Frequency (Hz)

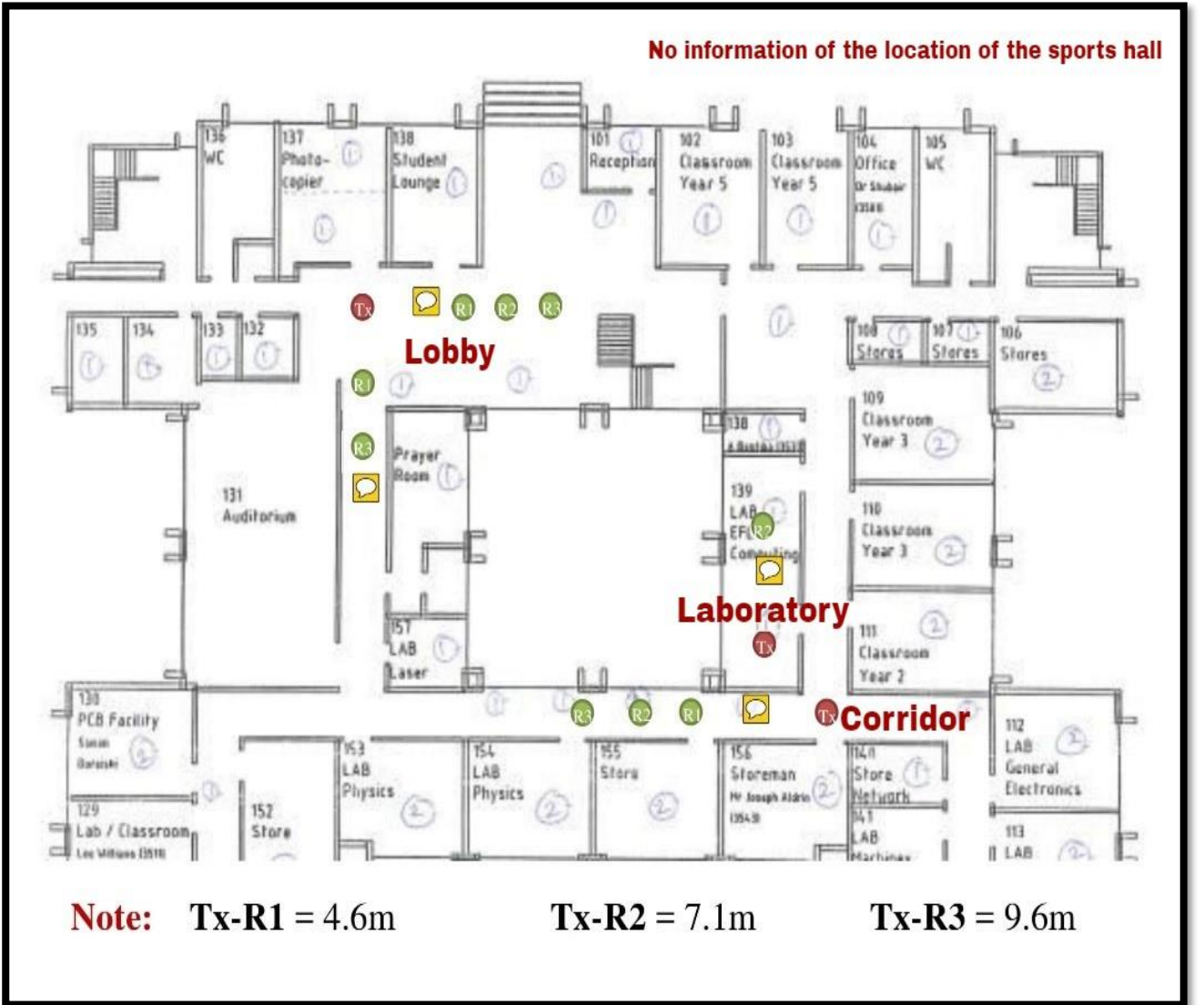


Figure 3.1 Floorplan of the area where the dataset is conducted

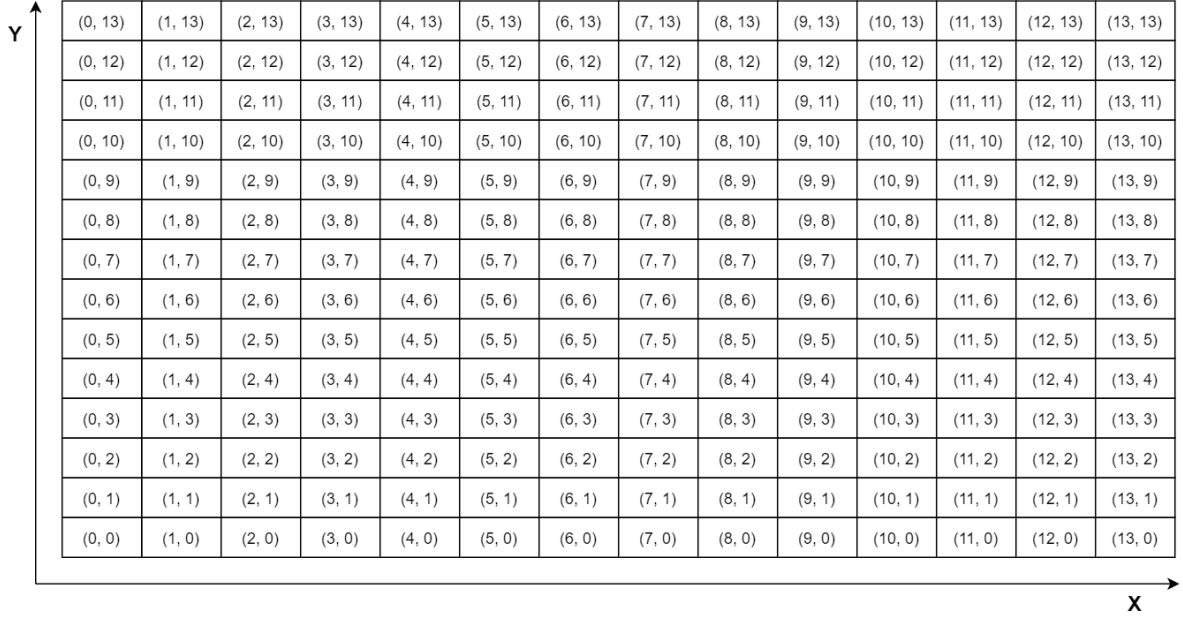


Figure 3.2 Grid area with each location's naming

3.2 Indoor Rooms Classification

The owners of this dataset's work [18, 19] classified signals collected from four rooms using four measurement metrics:

- Channel Impulse Response (CIR): a time-domain signal with a short duration.
- Channel Transfer Function (CTF): the fraction of the received signal $Y(f)$ to the sent signal $X(f)$ is denoted by CTF $H(f)$.
- Frequency Coherence Function: the autocorrelation of CTF.
- RSS: this is simply the signal's amplitude, and it has already been covered previously in **Section 2.1.1**.

The spatial correlation coefficient $\mathcal{C}^{(n)}$ of each can be calculated using the formula below:

Equation 3.1

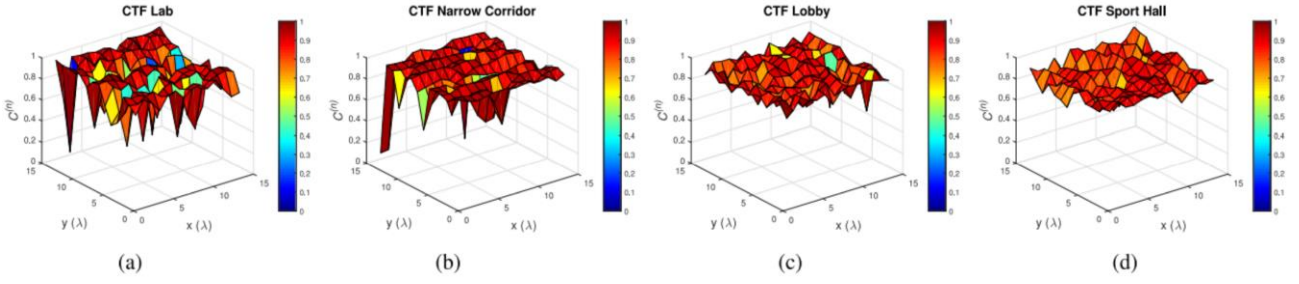
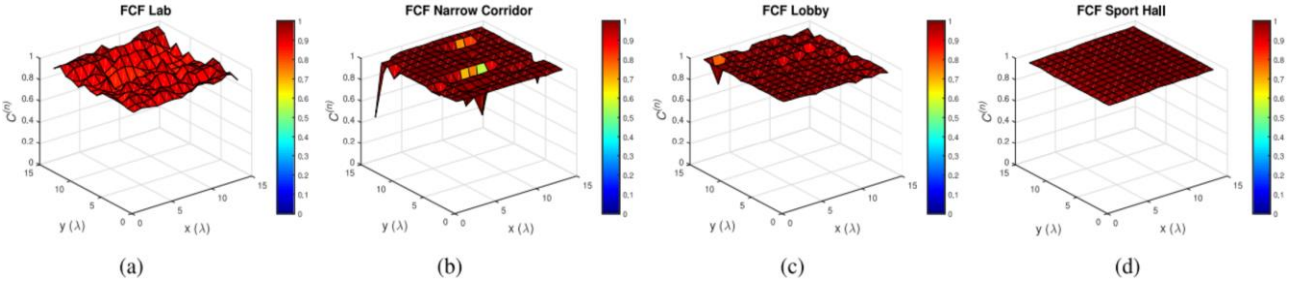
$$\mathcal{C}^{(n)} = \left| \frac{\langle S_r, S^{(n)} \rangle}{\|S_r\| \|S^{(n)}\|} \right|$$

where:

$S^{(n)}$: radio frequency signature associated with a particular place n

S_r : FCF or CTF's radio frequency signature at a reference location

As visualised in **Figure 3.3** and **Figure 3.4**, the level of CIR and CTF fluctuation is greatly dependent on the dispersion level of the room. The laboratory, for example, has the highest level of complexity and its CIR changes dramatically, while CIR of the sports hall (which is a relatively vacant area) varies slightly.


 Figure 3.3 $C^{(n)}$ of CTF [18]

 Figure 3.4 $C^{(n)}$ of FCF [18]

Subsequently, the dataset is divided into two sets, one for training and one for testing. Support Vector Machines, Decision Tree, and K-Nearest Neighbours are the three machine learning approaches that were examined. Finally, the results show that the most accurate model is the Weighted K-Nearest Neighbour (with $k = 1$) of the combination of CTF and FCF (99.3% accuracy). Furthermore, it shows that CTF and FCF are sufficient for this classification task since adding RSS to the combination does not boost accuracy.

The research in [21] proposed a new strategy that takes use of the mixed graph similarity algorithm (**Figure 3.5** fully depicts each step of the algorithm). The signals are first converted to the time domain using IFFT (the conversion process is discussed in **Section 2.4**). As a result, the data is transferred to the Horizontal and Natural Visibility Graphs. In the next step, the degree sequences are extracted from the graphs and fed to classification machine learning algorithms. Lastly, the outcome shows that K-Nearest Neighbours is the winner with 97.5% accuracy.

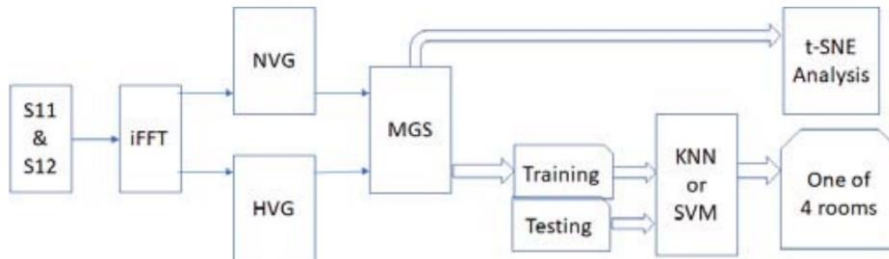


Figure 3.5 Flowchart of the algorithm proposed in [21]

In conclusion, the signals acquired from these four rooms are significantly different, with the variance in clutter level across the four rooms being the most critical determinant.

3.3 Similar Works

There are several studies that utilise UWB signals for indoor environments. For instance, [22] conducts an indoor placement experiment in a classroom with 12 attendees. A Decawave MDEK1001 UWB development kit is the primary tool for measurement, with four anchor tags in each corner of the ceiling and one tag for each attendee. Because the tags are placed on the ceiling, the test scenario is considered NOS (the signal travels directly from the anchor tag to the tag). There are two cases in this study. In the first case, the raw dataset from UWB is inserted into three clustering methods: K-Means Clustering, Mean-Shift Clustering, Fuzzy C-Means Clustering. The results from the clustering algorithm are filtered with Kalman Filter in the second case to obtain a better effect and improve the precision of the clustering methods. The outcomes are depicted in **Figure 3.6**, and it is clear from the charts that the filter has a positive impact. The average error of K-Means Clustering is decreased from 14.09cm to 9.27cm using the Kalman Filter. Besides, even though the difference is not notable, K-Means clustering surpasses the other algorithms in both cases.

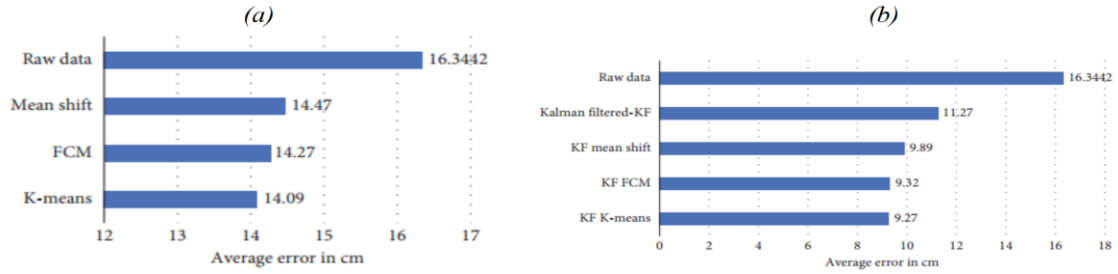


Figure 3.6 (a) Stage 1 (without Kalman Filter) (b) Case 2 (with Kalman Filter) [22]

The approach introduced in [23] uses TOA with UWB signals to solve the challenges of positioning vehicles in an underground basement. They selected TOA since they have done a quick test to compare its performance to RSS's. The result showed that TOA systems are more reliable, versatile, and power-efficient (this suggests that when locating with UWB signals, TOA should take priority). The primary experiment is carried out on a small car that is parked in a basement with three receivers on top of it. The transmitter is installed nearby at its position varies to collect the data from different distances and directions. The VNA is the equipment used to measure the signals (the same device used to obtain the dataset of this thesis project). With the signals, the distance from the transmitter to each receiver can be calculated. According to the results, one interesting point is that the calculated distances are always a little longer than the actual distances. For illustration, the actual distance between two points is 4.5 meters, whereas the calculated distance is 4.6125 meters (figures taken from Table 2 in [23]). One issue might be that the actual distances are not exact since they were measured with a ruler. Finally, with the distances calculated, the 2-D position of the car can be estimated using trigonometric.

Chapter 4

4 Indoor Environments Classification

4.1 Data Analysis and Preparation

As mentioned previously in **Section 3.1**, the system measured ten sweeps in total, meaning there are ten files of data for each location. Therefore, a straightforward technique has been applied to extract the data of S_{I1} and S_{2I} of each location by taking the mean value of those ten sweeps. Since the data are given in the form of complex number components (real part and imaginary part), the following formula has been used to calculate the magnitude of the two signals:

Equation 4.1

$$\text{magnitude (dB)} = 20 * \log_{10} * \sqrt{\text{real_part}^2 + \text{imaginary_part}^2}$$

As described in **Figure 4.1**, the magnitude of both S_{I1} and S_{2I} in four rooms varies considerably. The magnitudes of two signals fluctuate in a different notable range, which is a remarkable point to consider. In **Figure 4.1**, the magnitudes of S_{I1} fluctuate in the range from -15.5 dB to -16.0 dB (the highest peak is -13.5 dB, and the lowest peak is -16.0 dB). S_{2I} 's magnitudes, on the other hand, range from -55 dB to -45 dB in **Figure 4.1**, with the largest peak around -40 dB and the lowest peak surrounding -65 dB. These results convincingly demonstrate that signal S_{2I} is significantly weaker than signal S_{I1} in terms of signal strength.

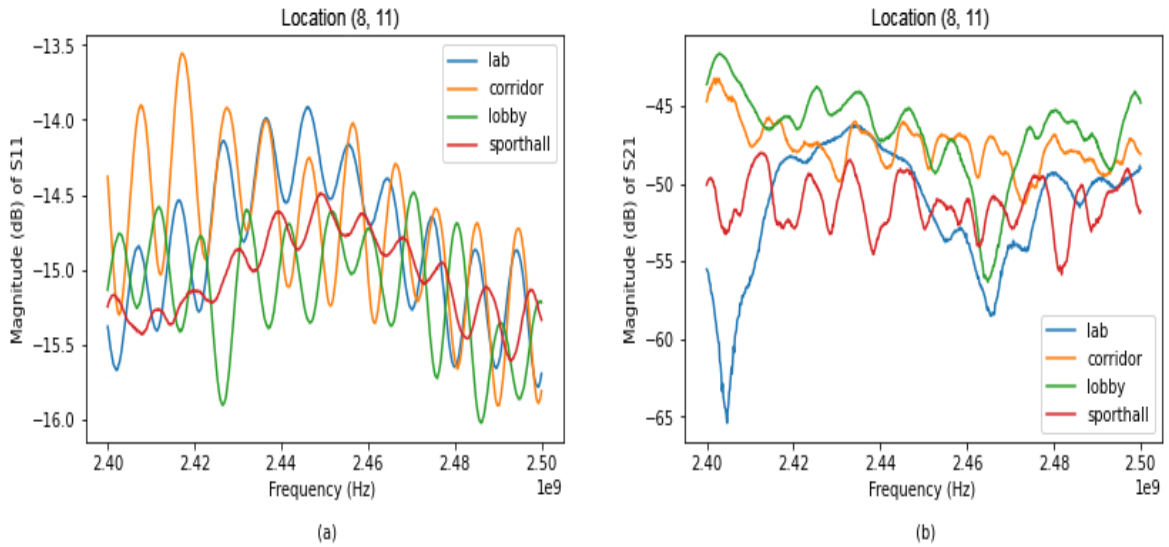


Figure 4.1 (a) Magnitude (dB) of S_{I1} (b) Magnitude (dB) of S_{2I}
at location (8, 11) of four rooms

Following that, the real and imaginary parts of the two signals are charted individually in Figure 4.2 for further analysis. It is evident that the real component of S_{11} in four rooms, as well as the imaginary part of S_{11} , have almost comparable patterns. On the contrary, the real and imaginary parts of S_{21} of each room follow an entirely different trend. The uncertainty of S_{21} can indicate that this signal is not stable nor powerful.

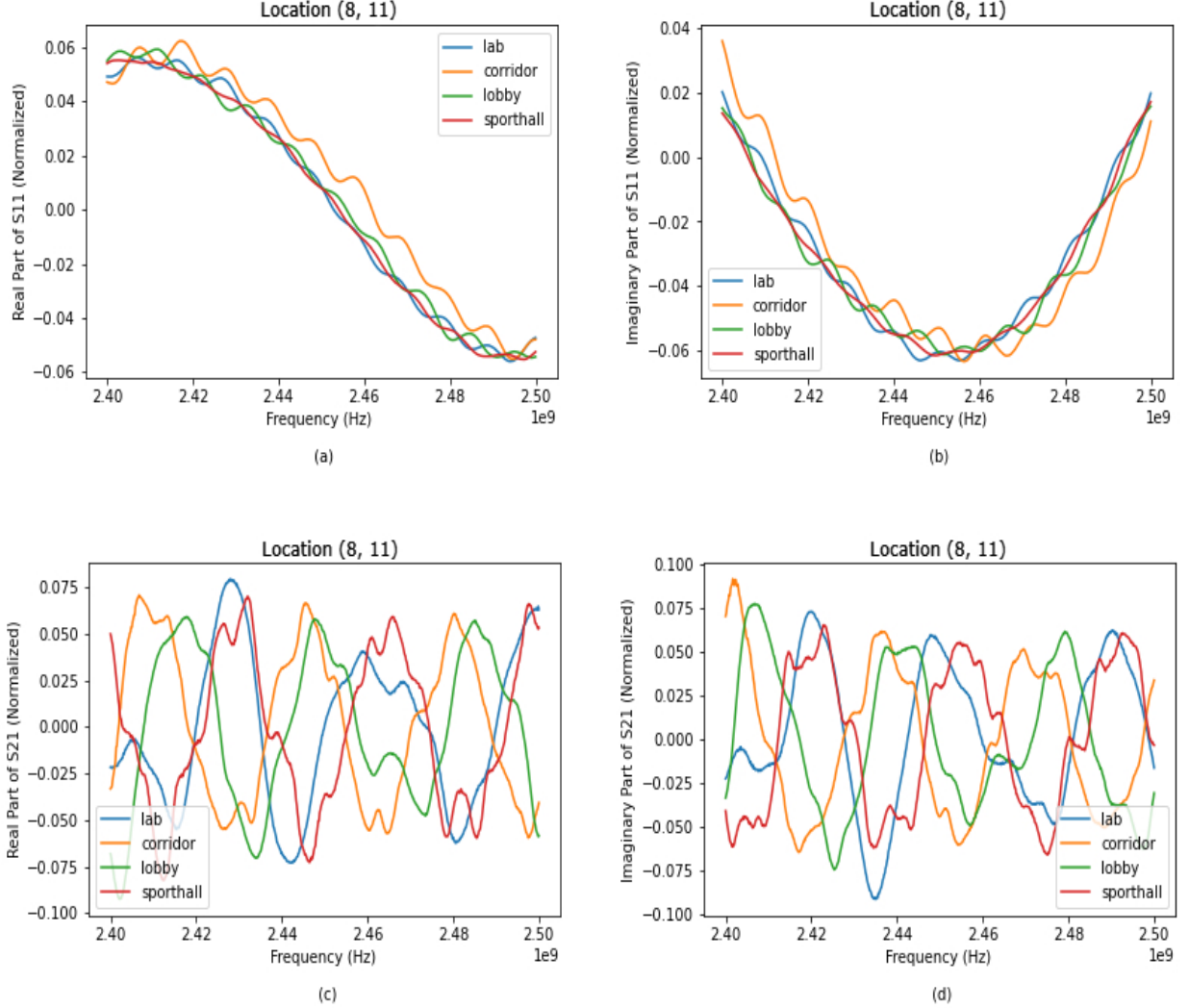


Figure 4.2 (a) Real part of S_{11} (b) Imaginary part of S_{11} (c) Real part of S_{21} (d) Imaginary part of S_{21} at the location (8, 11) in four rooms

Some can argue that the results may differ depending on the location since the analysis above is too specific to location 0208 (there are 196 locations, and location 0208 has been selected at random as an example in **Figure 4.1** and Figure 4.2). Therefore, **Figure 4.3** has been plotted using the signals from the laboratory in order to prove that the results are correct in all locations. As displayed in **Figure 4.3**, it is evident that for every location, the magnitude of S_{11} is always much stronger than the magnitude of S_{21} (the mean of S_{11} 's magnitude is around -14 dB, while the mean of S_{21} 's magnitude varies from -50 dB to -54 dB).

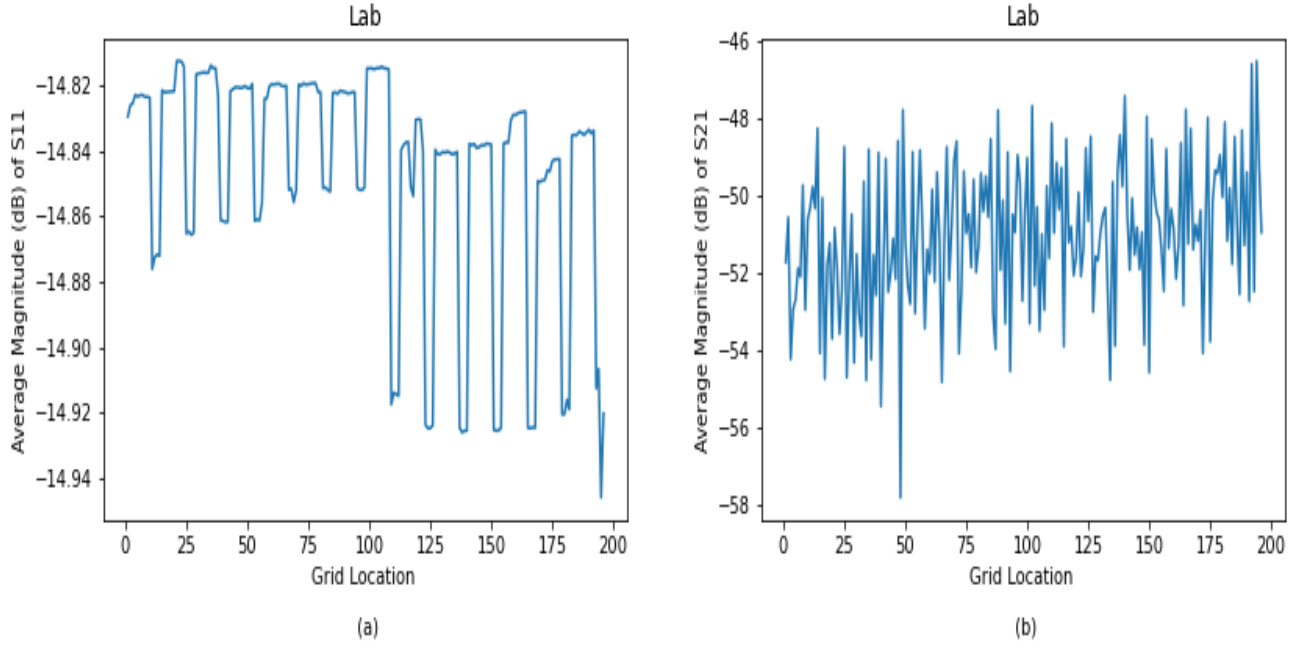


Figure 4.3 (a) Average magnitude (dB) of S_{11} (b) Average magnitude (dB) of S_{11} of 196 locations (signals from lab room)

As a result of the investigations in **Figure 4.1** and Figure 4.2, the following noteworthy conclusions arise:

- Because S_{11} is the reflected signal, it is significantly stronger than S_{21} (which is the transmission signal between two antennas). This explains why S_{11} is more powerful than S_{21} in **Figure 4.1**.
- As S_{21} is a weak signal, obstructions (such as room items and walls) between two antennas might cause it to be interfered with. As evidence, the real part and imaginary part of it oscillate dramatically and show no apparent pattern in Figure 4.2.
- The fact that S_{11} is robust and stable gives a hint that this signal might play an essential role in environments classification (the performances of S_{11} and S_{21} are compared in **Section 4.5**).

As stated earlier in **Section 3.1**, there are four rooms in this dataset, and each room contains the signal from 196 grid locations. Therefore, in total, there are $196 \times 4 = 784$ data points (observations). Each observation is a two-dimensional array with four columns (real part and imaginary part of S_{11} and S_{21}) and 601 rows (frequency sample points). **Table 4.1** describes the shape of this two-dimensional array in detail. Moreover, in order to operate with a variety of standard machine learning algorithms, this two-dimensional array should be flattened into a one-dimensional array with a length of $4 \times 601 = 2404$. To summarise, there are 784 data points and 2404 features, with each data point's label indicating which room it belongs to.

Table 4.1 Example of the two-dimensional array of each data point

	S ₁₁ _real	S ₁₁ _imaginary	S ₂₁ _real	S ₂₁ _imaginary
1	1.5081E-001	1.125E-001	1.501E-004	-3.452E-003
⋮	⋮	⋮	⋮	⋮
601	-1.643E-001	3.261E-002	2.332E-003	1.052E-003

4.2 Data Visualisation

In the field of machine learning, data visualisation is an essential step since it provides a better and wider understanding of the dataset. It is not difficult to visualise the dataset of two or three dimensions. However, in practice, the dimensions of the dataset can be thousands or even millions, so there must be some techniques to deal with this issue. One of the most common techniques for this situation is *dimension reduction*. This technique is used to convert a high-dimensional dataset into a low-dimensional dataset, as the name implies. Undoubtedly, this technique must ensure that the resulting dataset retains the original dataset's meaningful characteristics. Two *dimension reduction* methods that have been chosen for this dataset visualisation are Principal Component Analysis (PCA) and Uniform Manifold Approximation and Projection (UMAP).

PCA is an unsupervised approach for lowering data dimensionality, where each data point is projected onto only the first few principal components to produce a dataset with a lower dimension but still keeping as much variance as feasible. The first component could be characterised as a direction that maximises the projected data's variance. In other words, the first principal component involves the most variance and the later the component, the less it contains the variance. Overall, PCA offers three following major benefits:

- Because the dataset's size is small, it takes less time and resources to train.
- It removes noise by retaining only the most crucial information.
- It makes the visualisation of high-dimensional datasets achievable.

In order to improve the performance of PCA, the dataset is first standardised before being inserted into the algorithm. Since a three-dimensional dataset is possible to be visualised (humans cannot observe anything above three dimensions), three has been chosen as the number of PCA components for plotting in this dataset (the final result is displayed in

Figure 4.4). As shown in **Table 4.2**, three PCA components together represent 96.38% of the information and this is a satisfactory percentage for the purpose of visualising.

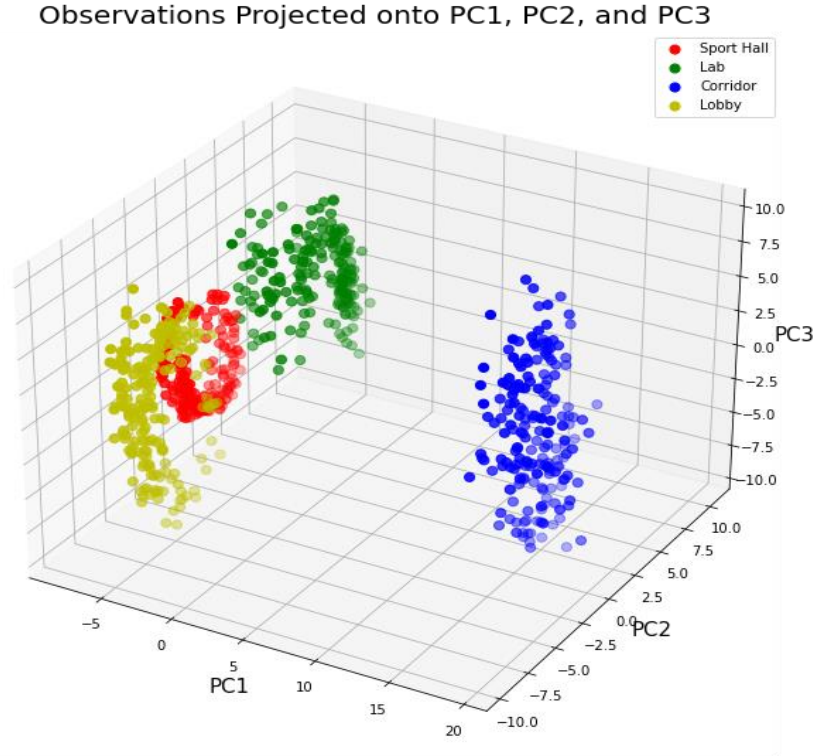


Figure 4.4 Observations projected onto three PCA components

Table 4.2 Explained variance of three PCA components

	PC1	PC2	PC3
Explained Variance	80.92%	13.83%	1.63%

UMAP was selected as an alternative approach to be compared with PCA in order to consolidate the results. UMAP, a brand-new technique proposed in [24], is fairly similar to t-Distributed Stochastic Neighbour Embedding (t-SNE), but it offers additional benefits. One of the main advantages, and one of the reasons UMAP has been chosen for this situation, is that it retains more global features in the final output than t-SNE [25]. There are four clusters (four rooms) in this dataset, and each cluster includes 196 points (196 locations); and the main goal is to determine whether those four clusters are clearly separated. Therefore, regarding this case, the global features are favoured over the local structure. There are two most widely used hyperparameters in UMAP:

- *n_neighbours*: the number of local neighbours that UMAP uses to learn the initial high-dimensional structure of data. Low values of this hyperparameter drive UMAP to focus on the local features, while higher values will push UMAP toward the global factors. For this problem, a high value (200) has been picked for *n_neighbours*.

- *min_dist*: the distance between points in the final result. In other words, it controls how densely UMAP can cluster points together. A low value of *min_dist* (0.1) was chosen for this case since the primary intention is clustering.

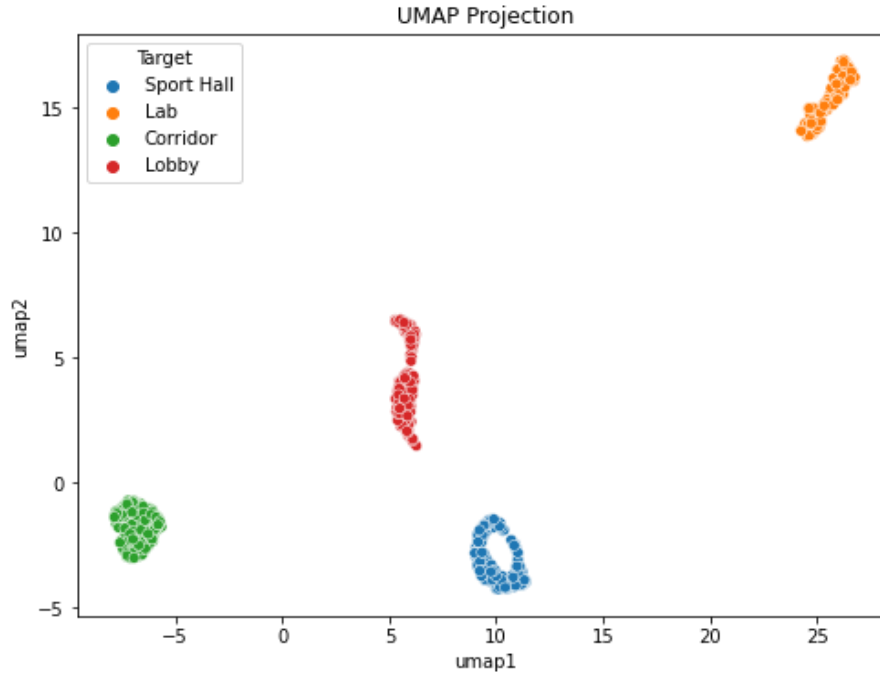


Figure 4.5 UMAP projection of the dataset

From **Figure 4.4** and **Figure 4.5**, it is evident that the four clusters are clearly separated, and there are no instances where points from one cluster are intermingled with those from another. These results partly reveal that if the classification algorithms are applied correctly, they may attain high accuracy.

4.3 Machine Learning Classification Algorithms

From the collected data samples, 70% of the dataset is utilised as the training set, while the remaining 30% will be used as the test set. Accordingly, the training set has $2404 * 70\% = 1683$ data points, and the test set contains $2404 * 30\% = 721$ data points. There are several classification algorithms available, and each has a different learning bias and is based on various assumptions, all of which might lead to a different aspect being discovered. For this classification problem, three representative algorithms have been selected:

1. K-Nearest Neighbours (KNN): KNN is one of the most basic and most often used algorithms for classification problems. One fundamental assumption in KNN is that: “similar data points stay in close proximity”. When a new observation is tested, KNN will classify it into a category (class) that contains observations that are most similar to it. k , which is the most vital hyperparameter of KNN, is the number of closest data points that KNN assesses as doing classification. Choosing the optimal k for this

algorithm is the critical step to achieve the best outcome, and one straightforward approach is to run KNN multiple times with different k then pick the one with the least error.

2. Support Vector Machine (SVM): The main objective of SVM is to shift all the data into a relatively high-dimensional space and then identify a hyperplane, which can successfully categorise the data points in that high-dimensional space. Even though SVM is popular for binary classification, it can still be applied to multiclass classification tasks by using the *one-vs-all* technique (isolating one class from the others and applying SVM repeatedly) [26]. SVM has the following advantages with respect to other classification algorithms:
 - It is memory efficient since it utilises a subset of the train set in the decision function, and this subset contains support vectors [27].
 - SVM is adaptable as different kernel functions can be used for different scenarios.
 - When working with high-dimensional datasets, SVM primarily performs correctly, which is valuable for this classification task because there are 2404 features.
3. Random Forest Classifier (RFC): RFC consists of multiple decision trees, which is the core component. Decision tree is a model with a tree-like structure in which each node represents a criteria evaluation on an attribute. Since a single decision tree is insufficient to deliver effective outcomes, RFC solves the problem by generating decision trees at random and combining the results to obtain a more accurate output. The most noticeable advantage of RFC is that it uses the *bagging* method to reduce data overfitting and hence enhance accuracy [28].

The three classification algorithms above are trained using the package named *scikit-learn* [29] in Python. This package is trendy because it contains helpful tools for several machine learning techniques. **Table 4.3** displays the accuracy as well as the configurations of each classification model (the default value is used for the hyperparameters that are not mentioned in **Table 4.3**).

Table 4.3 The configurations and accuracy of three classification models

	KNN	SVM	RFC
Configurations	$k = 3$ $weights = \text{'uniform'}$ $metric = \text{'euclidean'}$	$kernel = \text{'linear'}$	$n_estimators = 50$ $max_depth = \text{None}$
Accuracy (test set)	99.23%	99.49%	100%

From **Table 4.3**, it is clear that the accuracy of all three models is excellent. RFC is perfectly correct (100%), followed by SVM with an accuracy of 99.49%, and finally KNN with 99.23% accuracy. It is hardly unexpected that the outcomes are outstanding, given what was anticipated in **Section 4.2**. Even though SVM and KNN fall short of RFC in terms of performance, the difference in precision is negligible, and it is reasonable to conclude that these three algorithms are ideal for this classification task.

Since KNN achieves the lowest accuracy, it is worth being further investigated. In order to determine the relationship between the accuracy and the value of k , the algorithm is run with the value of k ranging from 1 to 50. **Figure 4.6** illustrates that the higher the value of k , the higher the error of the model. Therefore, the optimal option is to choose k between 1 and 3.



Figure 4.6 Error rate vs value of k in KNN

4.4 Deep Learning Neural Networks

In the last few years, there have been several advancements in the domains of data sciences and statistical analysis to categorise data. Elementary machine learning algorithms (such as KNN, SVM, RFC) and neural networks are some of the most effective data categorisation methods. Each approach has its own set of pros and limitations; however, when compared to others, neural networks are believed to perform well with complex cases, where normal machine learning algorithms perform poorly. Furthermore, neural networks have shown to be particularly effective in multiclass classification tasks, giving them an edge over classic SVM, which are better suited to binary data classification. As a result, even though regular machine learning models give perfect results in **Section 4.3**, neural networks are employed and discussed in this section for research reasons.

There are several types of neural networks, each of which is appropriate for a particular purpose or dataset. Regarding this dataset, although the signals are in the frequency domain, it is still considered as time series or a sequencing problem, in other words, and so Recurrent Neural Network (RNN) is the right match. The primary distinction between a conventional neural network and an RNN is that RNN has loops, which enable information to persist over time (**Figure 4.7** describes the architecture design of RNN). For example, at time t , besides calculating output h_t , it also updates its internal state and sends information of its state to the next time step. Another way to interpret RNN is to unroll it over time. Consider RNN as having numerous copies of the same network, with each block transmitting the message to the next one.

However, the standard version of RNN has an issue with vanishing gradients when facing long sequences. As a result, an improved RNN termed Long Short-Term Memory (LSTM) has been developed to address this issue. Generally, LSTM models are a subclass of RNN that are capable of learning and remembering across lengthy sequences of input data. As seen in **Figure 4.7**, the computation inside each block of LSTM is much more complex than RNN. The core objective of these calculations is to manipulate the information in the blocks selectively. The key computation of LSTM is called *gate*, and its role is to employ the *sigmoid* activation function to selectively add or remove input as it enters the block (*sigmoid* maps the input to the value between 0 and 1, which means it determines how much of the information passed to the gate should be retained).

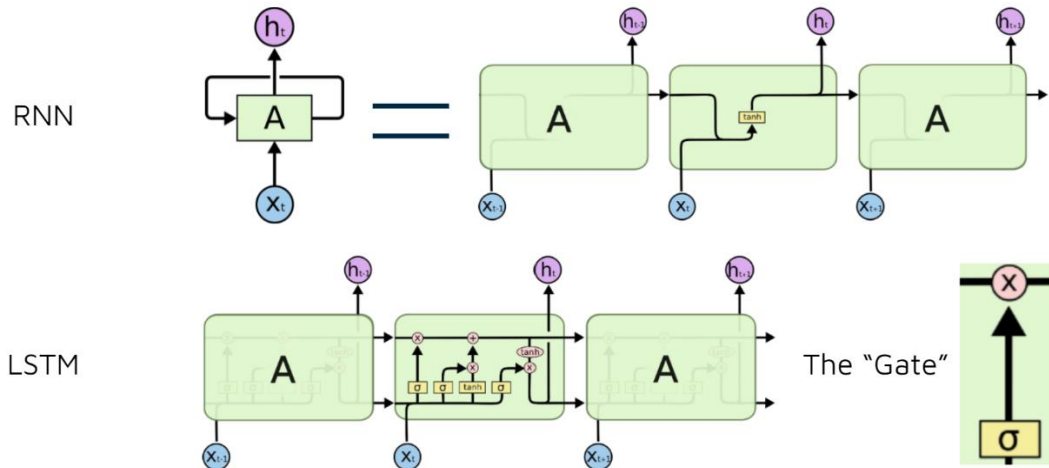


Figure 4.7 Comparison between RNN and LSTM

Because the input for this classification task is a two-dimensional array (**Table 4.1**), LSTM cannot be employed; however, versions of LSTM based on the convolutional principle may. Convolutional Neural Networks (CNN) have gained popularity in recent years due to their ability to handle two-dimensional input (Image processing is the most prevalent application). There are two most typical variations of LSTM that feature convolutional concept, and they are chosen for this classification task:

1. Convolutional Neural Network with LSTM (CNN-LSTM): This neural network combines CNN layers for feature extraction from input data with LSTM assistance for sequence prediction.

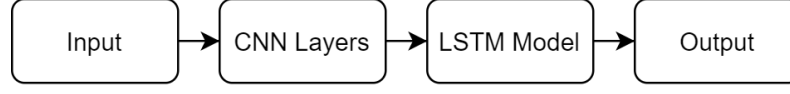


Figure 4.8 Flowchart of CNN-LSTM

2. LSTM Neural Network with Convolutional Operation: This neural network is a variant of the basic LSTM. The key difference is that it integrates the convolutional operation right inside each LSTM block.

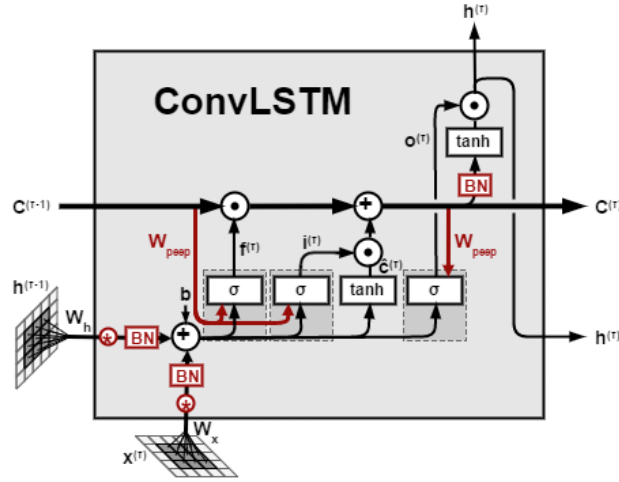


Figure 4.9 Architecture of a block of ConvLSTM

Below are the configurations of each model and they are referenced from [30] with the parameters and layers modified to suit this classification task (the models and the layers are created using Keras from Tensorflow [31]).

Configuration 4.1 Configuration of CNN-LSTM

```

1. output_length, features = train_y.shape[1], train_X.shape[2]
2. # reshape data into time steps of sub-sequences
3. steps, step_length = 6, 100
4. train_X = train_X.reshape((train_X.shape[0], steps, step_length, features))
5. test_X = test_X.reshape((test_X.shape[0], steps, step_length, features))
6. val_X = val_x.reshape((val_X.shape[0], steps, step_length, features))
7. # define model
8. model = Sequential()
9. model.add(TimeDistributed(Conv1D(kernel_size=3, filters=32, activation='relu'),
    input_shape=(None, length, features)))
10. model.add(TimeDistributed(Conv1D(kernel_size=3, filters=32, activation='relu')))
11. model.add(TimeDistributed(Dropout(0.5)))
12. model.add(TimeDistributed(MaxPooling1D(pool_size=2)))
13. model.add(TimeDistributed(Flatten()))
14. model.add(LSTM(100))
15. model.add(Dropout(0.5))
16. model.add(Dense(100, activation='relu'))
17. model.add(Dense(output_length, activation='softmax'))
18. model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
19. callback = EarlyStopping(monitor='val_accuracy', min_delta=1, patience=20)
20. model.fit(train_X, train_y, batch_size=64, epochs=100, validation_data=(val_X, val_y),
    callbacks=[callback])
    
```

Configuration 4.2 Configuration of ConvLSTM

```

1. output_length, features = train_y.shape[1], train_X.shape[2],
2. # reshape data into time steps of sub-sequences
3. steps, step_length = 6, 100
4. train_X = train_X.reshape((train_X.shape[0], steps, step_length, features))
5. test_X = test_X.reshape((test_X.shape[0], steps, step_length, features))
6. val_X = val_x.reshape((val_X.shape[0], steps, step_length, features))
7. # define model
8. model = Sequential()
9. model.add(ConvLSTM2D(kernel_size=(1,3), filters=32, input_shape=(steps, 1, length, features),
activation='relu'))
10. model.add(Dropout(0.5))
11. model.add(Flatten())
12. model.add(Dense(100, activation='relu'))
13. model.add(Dense(output_length, activation='softmax'))
14. model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
15. callback = EarlyStopping(monitor='val_accuracy', min_delta=1, patience=20)
16. model.fit(train_X, train_y, batch_size=64, epochs=200, validation_data=(val_X, val_y),
callbacks=[callback])

```

The original dataset is split into two sets, the first set (30%) is the test set, and the other set (70%) is used as input for the neural networks. Subsequently, K-Cross Validation (with $k = 5$) is applied on the second set to obtain the train set and the validation set. Some primary parameters and layers of both configurations are:

- The input must be separated into sub-sequences since these neural networks interpret the data as blocks. In order to be divided evenly, 601 frequency points are decreased to 601 points and then split into six sub-sequences of 100 frequency points.
- Dropout and Flatten layers are added to minimise *overfitting*.
- The Dense layers are added at the end to change the dimension of the vector from previous layers to the dimension of the desired output. In the last Dense layer, *softmax* is chosen as the activation function since the result should be read as a probability distribution.
- The validation data is used for the EarlyStopping option of the model. With this option enabled, the model will stop (even if the maximum number of epochs has not been reached) if the accuracy of the validation set does not increase 1% ($min_delta = 1$) for a certain number of epochs, which is the parameter *patience* set to 20.

As k is set to 5 in K-Cross Validation, each model is run five times and **Table 4.4** displays the model's accuracy for each iteration. Based on the results, it is safe to state that ConvLSTM is a better choice for this classification task when compared to CNN-LSTM. ConvLSTM gets 100% accuracy in four out of five iterations, but CNN-LSTM only achieves 100% accuracy in the two last iterations of the process. In terms of the mean value of the accuracy, CNN-LSTM has a 94.12%, whereas ConvLSTM has a 99.38%. The difference between the two models is little, and while ConvLSTM is superior, CNN-LSTM cannot be dismissed because of its high accuracy.

The learning/loss curves of CNN-LSTM and ConvLSTM are visualised in **Figure 4.10**. One interesting point is that, despite the fact that the maximum number of epochs is set to 200, both models stop short of that number (because of the EarlyStopping option that has been mentioned above). CNN-LSTM stops at nearly 50 epochs, whereas ConvLSTM terminates at around 55 to 70 epochs. This means that ConvLSTM is more expensive in terms of time, but the trade-off is that it has excellent accuracy. CNN-LSTM's learning and loss curves change dramatically in the first 20 epochs; then, the changes become minor in subsequent iterations. ConvLSTM's results show a similar trend, although its loss curves change significantly near the ending.

Table 4.4 Accuracy of CNN-LSTM and ConvLSTM

Iteration					
Model	1	2	2	4	5
CNN-LSTM	99.2%	76.8%	94.6%	100%	100%
ConvLSTM	100%	96.9%	100%	100%	100%

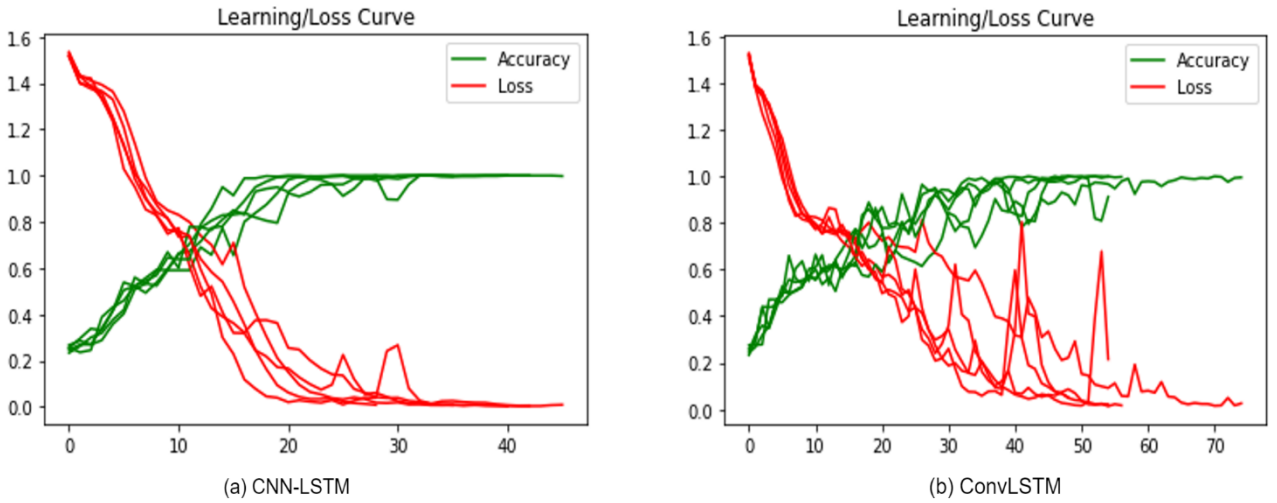


Figure 4.10 Learning/Loss Curves (a) CNN-LSTM (b) ConvLSTM

4.5 Results Comparison and Discussion

In general, the purpose of this classification task is to classify the signals from four different rooms, and it is similar to the work carried out in [18, 21], which has been covered in **Section 3.2**. The major difference is that instead of employing advanced metrics and methodologies like in those two articles, the classification job in this thesis project uses raw input (S_{11} and S_{21}). Despite the use of raw data, the outcome is equivalent to those of the other two articles.

According to **Section 4.3** and **Section 4.4**, both standard machine learning algorithms and deep neural networks perform almost flawlessly with this classification task. The outstanding results are not unexpected, as predicted in **Section 4.2** when the data was displayed using PCA. Therefore, it is reasonable to state that this classification task is not complex, and standard machine learning classifiers (such as KNN and SVM) are sufficient. Neural networks also show positive outcomes, but for this case, they are overkill (the apparent trade-off when employing neural networks is time). The outcome of this section plays a vital role in the success of the grid positioning algorithm, which is introduced in **Section 5**.

Finally, in order to compare the importance between S_{1I} and S_{2I} , the ConvLSTM model is selected to train each signal separately (previously, all the models and algorithms utilise both signals). The accuracies are listed in **Table 4.5**, and it is clear that S_{1I} is more valuable than S_{2I} since its accuracy is much higher. This result also reinforces the theory that S_{1I} is stronger because it is the reflected signal (covered in detail in **Section 4.1**). Nevertheless, combining the two signals yields the maximum performance in the end.

Table 4.5 Comparison between S_{1I} and S_{2I}

Model	Mean Accuracy		
	S_{1I}	S_{2I}	$S_{1I} + S_{2I}$
CNN-LSTM	87.26%	65.87%	94.12%

Chapter 5

5 Grid Location Positioning Algorithm

5.1 Frequency Domain and Time Domain Comparison

In the field of signal analysis, there is no doubt that the two most important variables are time (t) and frequency (f). Accordingly, signals are usually represented in the time domain or the frequency domain. In the time domain, the signal is displayed with regard to its amplitude versus time. In contrast, the signal is shown with respect to its amplitude versus frequency in the frequency domain (**Figure 5.1** illustrates an example of the two domains).

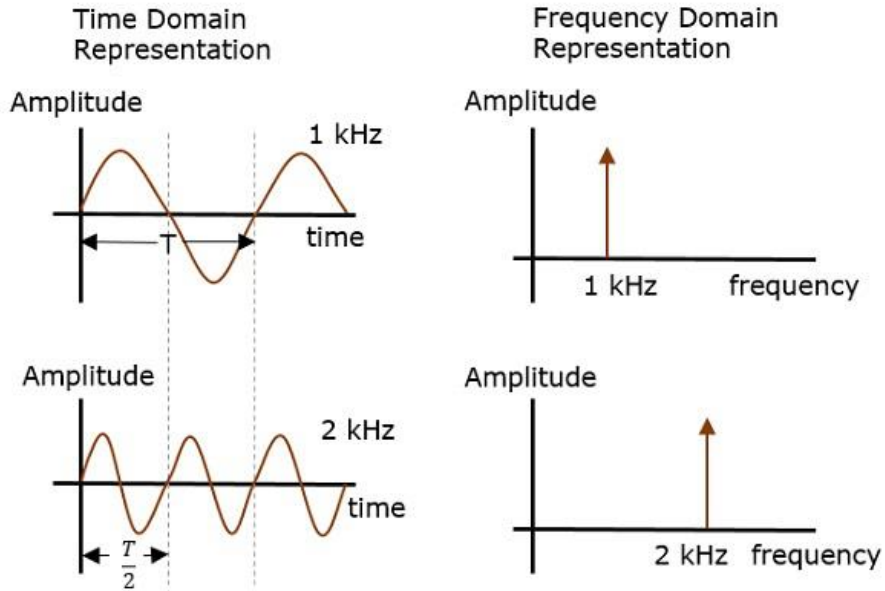


Figure 5.1 Two signals with different frequencies in the frequency domain and time domain [32]

1. Time domain: In the laboratory, the time domain of a signal is measured using an oscilloscope. According to [16], the time-domain representation $S(t)$ conveys the information about the signal's real presence, amplitude, and temporal evolution. It also depicts the distribution of the signal along the time axis with the start time and finish time. In other words, the graph of signal in the time domain can describe the process of how a signal varies over time. This characteristic of the time domain enables signal predictions and machine learning regression models.
2. Frequency domain: On the other hand, the frequency domain of a signal is evaluated by a spectrum analyser (or a signal analyser). Different from the time domain, the

frequency-domain representation shows how much of the signal appears within a range of frequency points. The analysis of the frequency domain is beneficial in situations where noise filtering and mixing are necessary.

The work in **Section 4** is carried out using a dataset in the frequency domain since the signal's characteristics in terms of frequency points are crucial in distinguishing the four interior settings (the research from [18] has already justified that frequency domain is excellent for this classification task). However, in order to perform the algorithm for grid site positioning, the data should be converted into the time domain. As stated earlier, the time domain contains the key information about the signal's strength and related properties over time. This information makes it suitable to carry out positioning with various techniques such as RSSI or TOA.

5.2 Step 1 - Time Domain Conversion

Because the signal data in this dataset was measured using a VNA, the signals obtained are in the frequency domain; hence, it must be converted to the time domain for positioning purposes. As explained in **Section 2.4**, the conversion process consists of the following steps so that the frequency-domain function follows Hermitian Symmetry:

1. Add zero-padding before the collected frequency range.
2. Reflect the complex conjugate of the S-parameters (including zero-padding) in the negative frequencies.
3. Apply IFFT to gather the real-valued time-domain function.

According to the dataset, there are $N = 601$ frequency sample points, ranging from f_{min} 2.4 GHz to f_{max} 2.5 GHz, and the related variables can be computed using **Equation 2.6**:

$$\text{Frequency step: } \Delta f = (f_{max} - f_{min})/N = 0.167 \text{ MHz}$$

$$\text{\# of zeros added: } N_0 = \frac{N}{(f_{max} - f_{min})} * f_{min} = 14424$$

$$\text{\# of points after reflecting complex conjugate: } N_{total} = (N + N_0) * 2 = 30050$$

$$\text{Maximum time point: } t_{max} = 1/\Delta f = 5988 \text{ ns}$$

$$\text{Time step: } \Delta t = 1/(\Delta f * N_{total}) = t_{max}/N_{total} = 0.2 \text{ ns}$$

The number of zero-padding is $N_0 = 14424$, and the total number of points after reflecting complex conjugate is $N_{total} = 30050$. In the time domain, the sample distance is $\Delta t = 0.2$ ns, calculated based on the frequency step Δf and N_{total} . The time pulse after applying IFFT ranges between 0 and $t_{max} = 5988$ ns.

The location (8, 11) from the lobby is selected to visualise the time-domain conversion results. **Figure 5.2** depicts the time-domain pulse from 0 to 5988 ns in (a), and the time-domain pulse is zoomed in from 0 to 100 ns for detailed examination in (b). The oscillating behaviour of the time-domain pulse following the *sinusoid* shape, which occurs after the frequency domain truncated function *rect*, can be noticed.

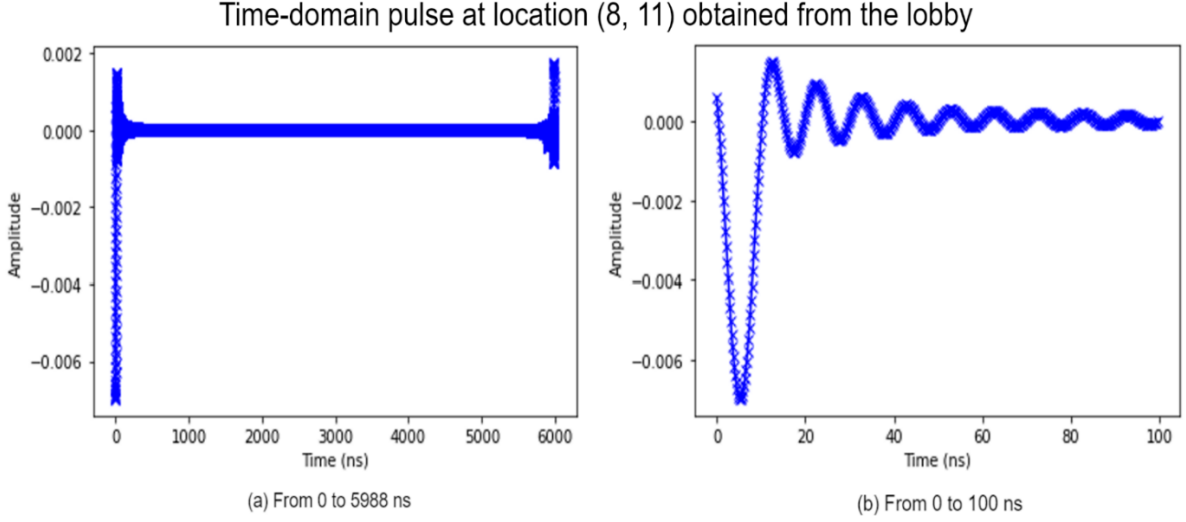


Figure 5.2 Converted time-domain signals (a) From 0 to 5988 ns (b) From 0 to 100 ns

The code snippet below describes the process in detail (the IFFT function is from the library *scikit-learn* [29]). Since the result is the real-valued time-domain function, each data point (location) is now a two-dimensional array with 30050 rows and 2 columns (S_{11} and S_{21}).

Algorithm 5.1 Algorithm to convert the S -parameters from frequency domain to time domain

```

1. def timeDomain(freq_dataset):
2.     # define variables
3.     freq_start = 2.4; # GHz
4.     freq_stop = 2.5; # GHz
5.     sample_points = 601
6.     freq_step = freq_stop - freq_start # 0.167 MHz
7.     num_of_zeros = math.ceil((sample_points/freq_step) * freq_start) # 14424
8.     # begin conversion
9.     time_dataset = {}
10.    for room in rooms:
11.        room_data = {}
12.        for location in locations:
13.            # Retrieve frequency-domain singals
14.            s11_re, s11_im, s21_re, s21_im = read_freq_data(area, location)
15.            s11_complex = s11_re + 1j * s11_im
16.            s21_complex = s21_re + 1j * s21_im
17.            # Convert S11
18.            padded_data = np.concatenate([np.zeros(math.floor(num_of_zeros)), s11_complex])
19.            conj_data = np.concatenate([np.flip(np.conj(padded_data)), padded_data])
20.            final_data = np.concatenate([[0+0j], conj_data])
21.            s11 = np.fft.ifft(final_data);
22.            # Convert S21
23.            padded_data = np.concatenate([np.zeros(math.floor(num_of_zeros)), s21_complex])
24.            conj_data = np.concatenate([np.flip(np.conj(padded_data)), padded_data])
25.            final_data = np.concatenate([[0+0j], conj_data])
26.            s21 = np.fft.ifft(final_data);
27.            # Combine both S11 and S21
28.            room_data[location] = np.array([s11.real, s21.real]).T
29.        time_dataset[area] = room_data
30.    return time_dataset
    
```

5.3 Step 2 – Divide into Zones

Conventionally, for positioning purposes, regression models are usually used instead of classification models. However, regarding this dataset, it does not provide statistics about the distances between each grid location and the transmitters/receivers (despite the fact that the area's design is known, no measurements have been given). Therefore, one of the possible options is that in this second step, the 14×14 grid area is divided into smaller zones so that in the next step, machine learning classifiers will be used to predict which zone a location belongs to. A zone is a group/set of locations that share similar characteristics. This step is where the signal metrics, which have been introduced in **Section 2.1**, are applied. TOA, RSS, and AOA are chosen as the three metrics since they have been proven to be effective in various papers, and they will be utilised as a criterion when dividing into zones.

5.3.1 Step 2a – Divide into Zones based on TOA and RSS

Because the time domain expresses the signal's amplitude with time, TOA and RSS can be employed based on this feature. As a result, this step uses an unsupervised machine learning model to conduct zone division based on raw time-domain data. An unsupervised method is used for datasets without pre-assigned labels (output variables) to discover/learn the pattern by itself. Clustering is the most typical use of unsupervised learning, in which the algorithms look for features that are similar across data points and use that knowledge to classify the data. This zone division task is also a clustering problem, and the unsupervised algorithm's objective is to detect the TOA/RSS characteristic and then arrange locations into zones.

There are several popular unsupervised machine learning algorithms, and the one selected for this case is K-Means Clustering (shown to work effectively with UWB signals [22]). As there are not many computations within, the key benefit of it is speed. This algorithm consists of four steps:

1. Choose the k number of classes and produce the starting centre points at random (known as centroids). The most obvious difficulty in using the algorithm is deciding on the appropriate value for k .
2. The distances between the point and the centroids are used to determine which cluster that point belongs to.
3. Calculate new centroids by taking the mean of each cluster's data points.
4. Repeat step 1 to step 3 until the centroids remain unchanged (or until it reaches the maximum number of iterations set by the user).

K-Means Clustering is trained using scikit-learn [27] with the following parameters:

- $k = 8$: both the number of clusters (in this case, zones) and the number of centroids. After multiple testing with various values for k , the most optimum value is determined to be 8. The rationale for this is discussed in **Section 5.6**.
- $max_iter = 300$: the maximum number of iterations.
- $random_state = 0$: K-Means Clustering randomly initialises the centroids, meaning that the result differs each time. The random number generator is seeded by this parameter when it is set to a fixed integer; hence, the same result is produced across several runs. In other words, this parameter is specified to ensure that the same set of zones are used throughout the study. Initially, the algorithms were tested with several seeds, and the results showed little variation; therefore, one value was chosen to assure consistency.

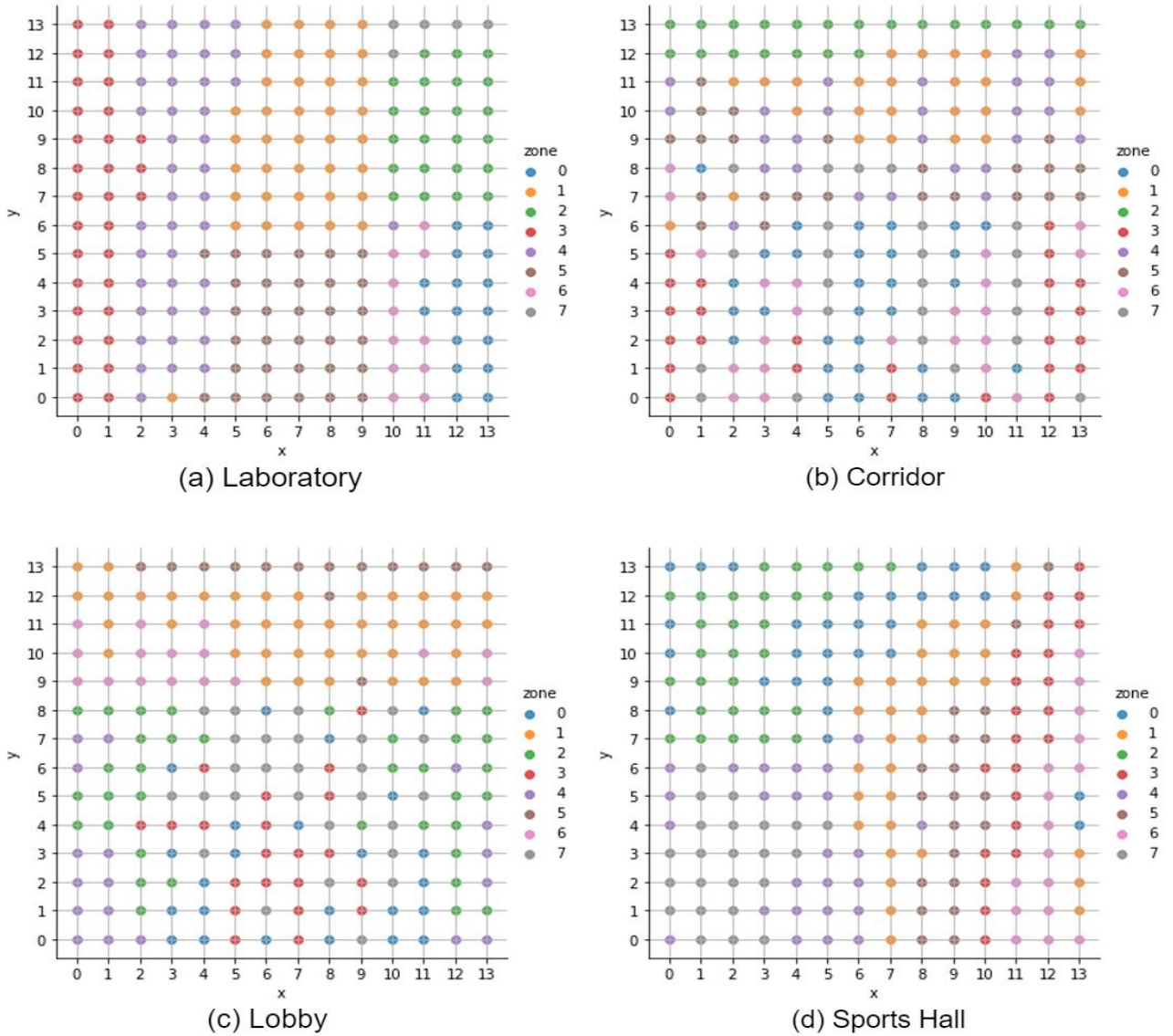


Figure 5.3 Four combinations of zones after applying Step 2a on the signals from four rooms
 (a) Laboratory (b) Corridor (c) Lobby (d) Sports Hall

Finally, since the signals from 196 locations are measured from four different rooms, this step can be applied four times results in four combinations of zones. Each combination of zones is visualised in **Figure 5.3**.

5.3.2 Step 2b – Divide into Zones based on AOA

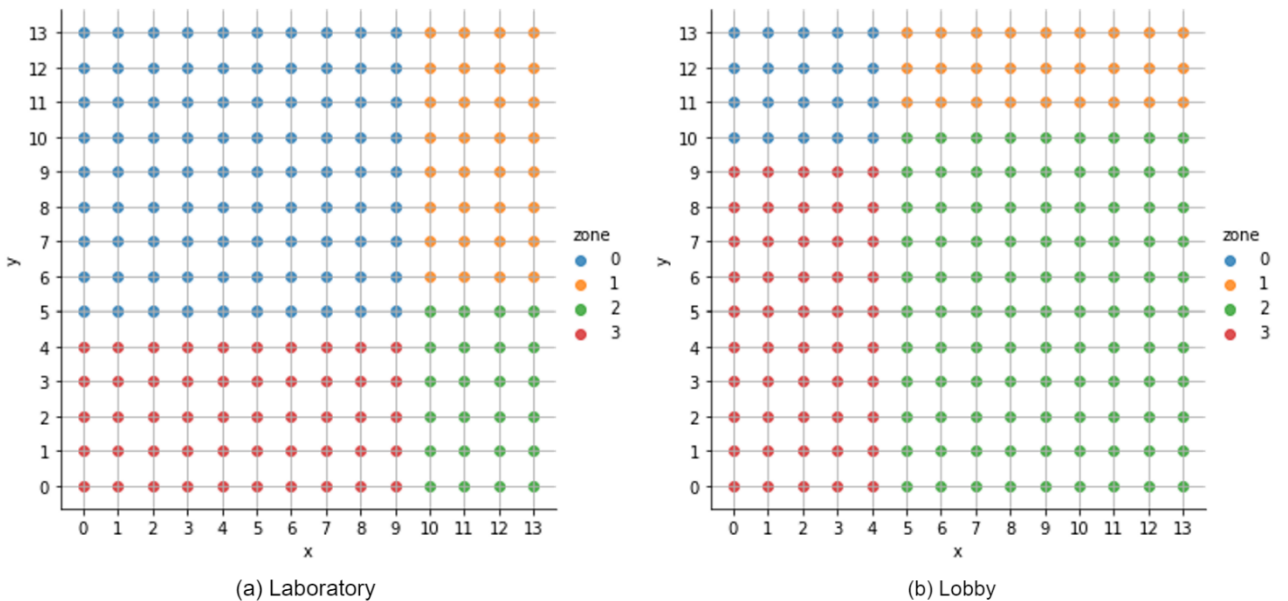
Instead of using TOA and RSS, Step 2b utilises AOA for division. This technique consists of two steps:

1. Based on the blueprint of the floor (**Figure 6.1**) and the grid numbering (**Figure 6.3**), estimate the position of the room in the grid area.
2. Subsequently, the area is divided into zones with respect to the room. In this case, the number of zones is selected to 4 (similarly to k in **Step 2a**, the justification for this value is detailed in **Section 5.6**). **Table 5.1** lists the description of each zone.

Table 5.1 Description of each zone in Step 2b

1 st zone	2 nd zone	3 rd zone	4 th zone
Grid locations at 270° to 0°	Grid locations at 0° to 90°	Grid locations at 90° to 180°	Grid locations at 180° to 270°

This step requires determining the location of the room within the grid area, and because the dataset does not include the location of the sports hall, this step can be applied to only three rooms: the lobby, the laboratory, and the corridor. The three combinations of zones are depicted in **Figure 5.4**. Furthermore, due to the fact that the laboratory and corridor are located adjacent to one another in the floor plan, their combination of zones is reasonably identical.



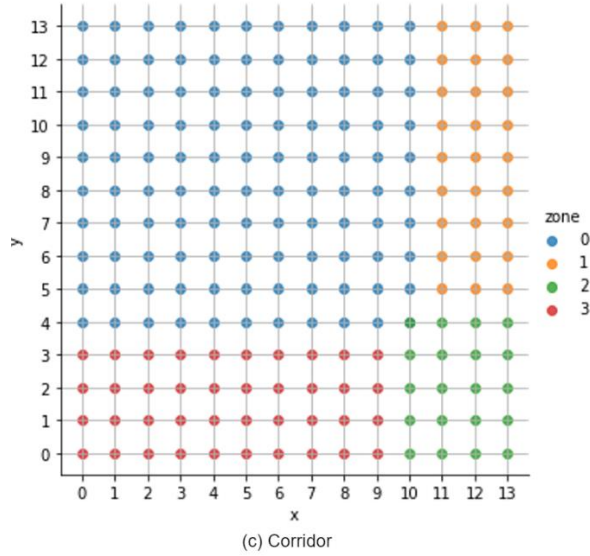


Figure 5.4 Three combinations of zones after applying Step 2b on the signals from three rooms
 (a) Laboratory (b) Lobby (c) Corridor

5.4 Step 3 – Zones Classification

With the time-domain signals from **Step 1** and seven combinations of zones from **Step 2**, the third step is where the machine learning classifiers perform zone classification. The inputs are 196 grid locations, each holding time-domain data (a two-dimensional array of shapes (30050, 2)). The seven combinations of zones, on the other hand, are employed as labels (outputs) for the classification algorithms. K-Cross Validation with $k = 5$ is used to divide 196 grid points into train and test sets. The reasons why K-Cross Validation is an excellent option in this case are:

- When compared to other splitting strategies, it produces a model that is less biased.
- Because this is a multi-class problem (there are four zones in the **Step 2b** zone combinations and eight zones in the **Step 2a** zone combinations), using the traditional train-test split technique is inadequate and ineffective, as there will be instances where data points for a class are missing from the train set but are present in the test set, or vice versa. Therefore, K-Cross Validation is a great alternative for ensuring that every location appears in both the train set and the test set.
- In this case, with $k = 5$, the models are tested five times with five distinct test sets to examine the accuracy of the models with every location. The greatest outcome is that the accuracy is consistent in all five folds. However, in a slightly different situation, one or two folds might experience poor accuracy in comparison to others.
- With K-Cross Validation, it is easier to perform hyperparameter tuning (choosing k in KNN as an example).

Because there are seven combinations of zones (or seven sets of labels), the models are trained and tested seven times with the data that corresponds to each zone combination. For example, for the two zones combinations from the laboratory in **Step 2a** (**Figure 5.3 (a)**) and **Step 2b** (**Figure 5.4 (b)**), the time-domain data obtained from the laboratory is used as the input. The entire process of this step is illustrated in **Figure 5.5**.

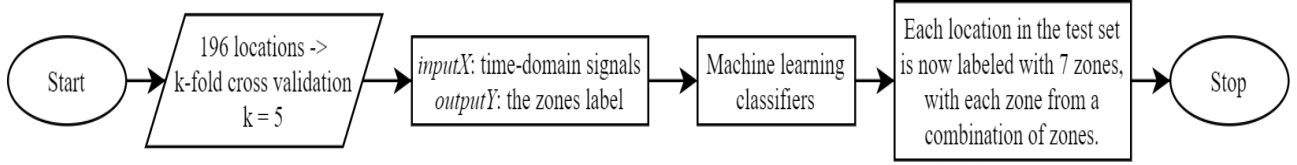


Figure 5.5 Flowchart of Step 3

As a result, each location in the test set is labelled with seven zones, with each zone from a combination of zones. For example, for illustration purposes, the grid point (8, 3) was picked, and the results are shown in **Figure 5.6** (assuming the classification models are correct). The following is an explanation of the results:

- The grid point (8, 3) is labelled as zone 5 from the laboratory's zones combinations from **Step 2a**.
- The grid point (8, 3) is labelled as zone 7 from the corridor's zones combinations from **Step 2a**.
- The grid point (8, 3) is labelled as zone 3 from the lobby's zones combinations from **Step 2a**.
- The grid point (8, 3) is labelled as zone 1 from the sports hall's zones combinations from **Step 2a**.
- The grid point (8, 3) is labelled as zone 3 from the laboratory's zones combinations from **Step 2b**.
- The grid point (8, 3) is labelled as zone 2 from the lobby's zones combinations from **Step 2b**.
- The grid point (8, 3) is labelled as zone 3 from the corridor's zones combinations from **Step 2b**.

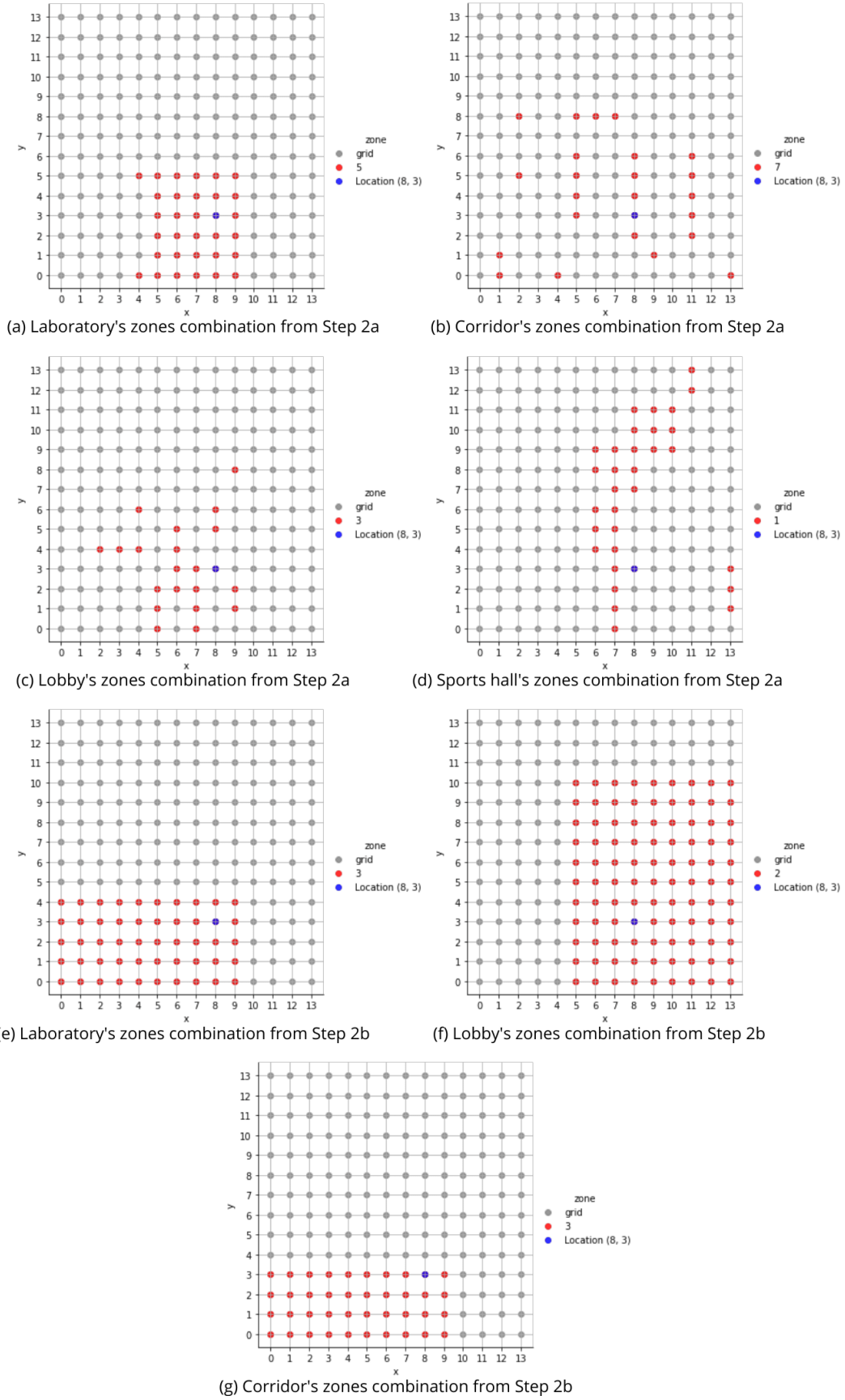


Figure 5.6 The labels of the location (8, 3) after Step 3

- (a) Laboratory's zones combination from Step 2a (b) Corridor's zones combination from Step 2a
(c) Lobby's zones combination from Step 2a (d) Sports hall's zones combination from Step 2a
(e) Laboratory's zones combination from Step 2b (f) Lobby's zones combination from Step 2b
(g) Laboratory's zones combination from Step 2b

5.5 Step 4 – Combine Zones

Following **Step 3**, each location in the test set is labelled with seven zones, with each zone from a combination of zones. The last step is to combine each location's zone labels to reveal the final location prediction. The most obvious and quickest way is to take the intersection of the zone labels. For instance, the intersection of zone labels of the location (8, 3) (**Figure 5.6**) is the location (8, 3), and it is considered as a correct prediction. However, the machine learning classifiers in **Step 3** are not guaranteed to have 100% accuracy, so the intersection of the zone labels can be empty, resulting in a false prediction. Therefore, simply taking the intersection is a negative approach, and an algorithm is devised to solve this problem. Moreover, one important point here is that because the main idea is taking the intersection, it is not meaningful if the zone labels are too similar. Because the combinations of zones of laboratory and corridor of **Step 2b** are too identical (as previously stated), their zone labels are similar (**Figure 5.6 (e) and (g)**). Accordingly, the zone label from the corridor's zones combination from **Step 2b** is omitted, which means that each location now is only labelled with six zones.

The primary idea of the algorithm is to find the largest subset of zone labels with a non-empty intersection. The algorithms can be described as follows:

1. Assume that six zone labels are named a, b, c, d, e, f .
2. The largest subset size is 6 and there is only one subset that has this size (the subset that contains all six zone labels).
3. If the intersection of this subset is not empty, it is used as the final output.
4. If the intersection of this subset is empty, decrease the subset size to 5 and find all the subsets that have this size (because the initial set has six labels, 6 is the only size that only one subset satisfies). To illustrate, there are six subsets that have the size equals to 5 ($[a, b, c, d, e]$, $[b, c, d, e, f]$, $[a, c, d, e, f]$, $[a, b, d, e, f]$, $[a, b, c, d, f]$, $[a, b, c, d, f]$). Certainly, only one subset can be used as the output, and if there are many subsets with a non-empty intersection, there are two strategies to choose the best one. The first strategy is to choose the subset that has the smallest intersection, while the second is to pick the one with the largest intersection (the trade-off between these two strategies is discussed in **Section 5.6** since they have an impact on the result).
5. If there every subset has a non-empty intersection, continue decreasing the subset size by 1 and repeat the previous step until the result is found.

The algorithm is also explained in **Figure 5.7**, and the code is provided in **Algorithm 5.2** (the algorithm is programmed slightly different from **Figure 5.7**, but the idea is the same).

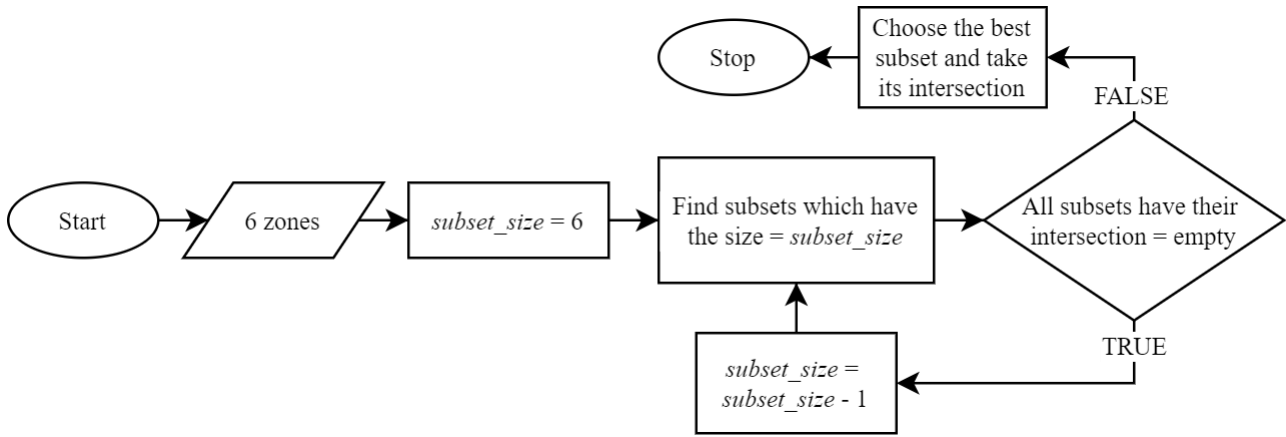


Figure 5.7 Flowchart of the algorithm of Step 4

Algorithm 5.2 Algorithm to combine the zone labels

```

1. def combine_labels(zones, strategy):
2.     distinct_locations_with_zones = {}
3.     for i in range(len(zones)):
4.         for location in zones[i]:
5.             if (location in distinct_locations_with_zones):
6.                 distinct_locations_with_zones[location].append(i)
7.             else:
8.                 distinct_locations_with_zones[location] = [i]
9.
10.    zones_label = []
11.    for value in distinct_locations_with_zones.values():
12.        if (value not in zones_label):
13.            zones_label.append(value)
14.
15.    max_size = 0
16.    for value in zones_label:
17.        if len(value) > max_size:
18.            max_size = len(value)
19.
20.    largest_zones_label = []
21.    for value in zones_label:
22.        if len(value) == max_size:
23.            largest_zones_label.append(value)
24.
25.    intersections = []
26.    for value in largest_zones_label:
27.        intersection = locations
28.        for i in value:
29.            intersection = list(set(intersection)&set(zones[i]))
30.        intersections.append(intersection)
31.
32.    if (strategy == 'min'):
33.        index = 0
34.        min = 196
35.        for value in intersections:
36.            if len(value) < min:
37.                min = len(value)
38.                index = intersections.index(value)
39.
40.    return intersections[index]
41.    else:
42.        index = 0
43.        max = 0
44.        for value in intersections:
45.            if len(value) > max:
46.                max = len(value)
47.                index = intersections.index(value)
48.
49.    return intersections[index]
    
```

5.6 Results and Discussion

Table 5.2 displays the accuracy of the machine learning classifiers that are used in **Step 3** for zones classification. All seven combinations of zones are trained with three classifiers: KNN, SVM, and RFC; and the results indicate that KNN matches the best with four from **Step 2a**, whereas RFC is the finest selection for three zone combinations from **Step 2b**. It is clear that the classifiers perform effectively with this task since the accuracies are almost over 90%, except for the accuracy of the corridor from **Step 2b** (only around 70%). Consequently, removing its label in **Step 4** is rational, besides the fact that it is too similar to the laboratory's zone combination. The performance of the classifiers here is crucial since it significantly influences the results generated in **Step 4**.

Table 5.2 Results of Zones Classification (Step 3)

Model	Zone Combinations	Fold				
		1	2	3	4	5
KNN	Laboratory (Step 2a)	100.000%	92.308%	87.179%	97.436%	89.744%
	Corridor (Step 2a)	97.000%	92.308%	97.436%	100.000%	97.436%
	Lobby (Step 2a)	92.500%	87.179%	97.436%	92.308%	89.744%
	Sports Hall (Step 2a)	95.000%	97.436%	97.436%	100.000%	89.744%
RFC	Laboratory (Step 2b)	100.000%	97.436%	94.872%	100.000%	94.872%
	Lobby (Step 2b)	100.000%	94.872%	97.436%	97.436%	100.000%
	Corridor (Step 2b)	69.231%	69.231%	74.359%	66.667%	71.795%

1. Location: (7, 3) - Correct Prediction: [(6, 4), (7, 3)]
2. Location: (13, 5) - Correct Prediction: [(13, 5)]
3. Location: (5, 11) - Correct Prediction: [(5, 11)]
4. Location: (7, 1) - Correct Prediction: [(7, 1), (7, 0)]
5. Location: (8, 9) - Correct Prediction: [(8, 9), (8, 10)]
6. Location: (3, 6) - Correct Prediction: [(3, 6)]
7. Location: (9, 4) - Correct Prediction: [(9, 4)]
8. Location: (3, 9) - Correct Prediction: [(3, 9), (4, 9)]
9. Location: (5, 4) - False Prediction: [(5, 3)]
10. Location: (1, 5) - Correct Prediction: [(1, 5)]
11. Location: (8, 0) - False Prediction: [(9, 1), (9, 2)]
12. Location: (8, 12) - False Prediction: [(7, 12), (9, 12), (6, 11), (7, 11)]
13. Location: (0, 8) - False Prediction: [(1, 8)]
14. Location: (11, 4) - Correct Prediction: [(11, 4)]
15. Location: (3, 2) - Correct Prediction: [(3, 2), (2, 1)]
16. Location: (1, 9) - Correct Prediction: [(1, 9), (2, 9), (0, 9)]
17. Location: (8, 6) - Correct Prediction: [(8, 6)]
18. Location: (6, 4) - Correct Prediction: [(6, 4), (7, 3)]
19. Location: (1, 3) - Correct Prediction: [(0, 1), (1, 3), (0, 3), (1, 2), (0, 2)]
20. Location: (0, 10) - Correct Prediction: [(0, 11), (0, 10)]
21. Location: (10, 8) - Correct Prediction: [(10, 8)]
22. Location: (13, 11) - Correct Prediction: [(13, 12), (13, 11)]
23. Location: (11, 9) - Correct Prediction: [(12, 10), (11, 9)]
24. Location: (10, 3) - Correct Prediction: [(10, 2), (10, 3)]
25. Location: (10, 13) - Correct Prediction: [(10, 13)]
26. Location: (1, 13) - Correct Prediction: [(0, 13), (1, 13)]
27. Location: (9, 1) - Correct Prediction: [(9, 1)]
28. Location: (9, 3) - Correct Prediction: [(9, 3)]
29. Location: (0, 5) - Correct Prediction: [(0, 5)]
30. Location: (12, 7) - Correct Prediction: [(12, 8), (12, 7)]
31. Location: (0, 1) - Correct Prediction: [(0, 1), (1, 3), (0, 3), (1, 2), (0, 2)]
32. Location: (11, 7) - Correct Prediction: [(11, 7)]
33. Location: (13, 3) - Correct Prediction: [(13, 2), (13, 3)]
34. Location: (2, 4) - Correct Prediction: [(2, 4)]
35. Location: (0, 0) - Correct Prediction: [(0, 0)]
36. Location: (4, 13) - Correct Prediction: [(4, 13), (3, 13)]
37. Location: (7, 10) - Correct Prediction: [(6, 10), (7, 10)]
38. Location: (7, 11) - Correct Prediction: [(7, 12), (9, 12), (6, 11), (7, 11)]
39. Location: (11, 8) - Correct Prediction: [(11, 8)]
40. Location: (8, 11) - Correct Prediction: [(8, 11)]

Figure 5.8 The prediction of the algorithm for each location in the test set (I^{st} fold)

Table 5.3 Three criteria for evaluating the performance of the grid positioning algorithm

	Fold				
	1	2	3	4	5
Accuracy	90.000%	82.051%	82.051%	92.307%	79.487%
Mean size of all predictions	1.800	1.820	1.769	1.692	1.384
Root-mean-square error	0.589	1.314	0.897	0.806	0.891

The main drawback of this grid positioning algorithm is that it does not guarantee that the prediction is a single-location prediction. For example, in **Figure 5.8**, line 2 is a single-location prediction, while line 1 is a multiple-location prediction (there are two locations in this

prediction). Therefore, the correct predictions are those that are either the actual location or contain the actual location, and the accuracy is the percentage of the correct predictions over all predictions. The accuracy is great throughout five folds, and the significant point is that it does not change remarkably (it fluctuates between 80% and 90%, **Table 5.3**). Some may argue that the method is overfitting because it is only trained and evaluated on the dataset's grid positions. However, this is not a problem because the dataset already contains all grid points on that floor, meaning that there are no missing locations.

As the algorithm does not guarantee single-location prediction, the size of the predictions should be examined. Assuming one location is one unit, the mean size of all predictions is outlined in **Table 5.3**. It is clear that the smaller this value, the more reliable the algorithm, because if the size of a prediction is too great, even if it is correct, that prediction is meaningless (**Figure 5.9** illustrates an example of this case).

(5, 8)	(6, 8)	(7, 8)	(8, 8)	<div>Prediction</div> <div>Actual Location</div>
(5, 7)	(6, 7)	(7, 7)	(8, 7)	
(5, 6)	(6, 6)	(7, 6)	(8, 6)	

Figure 5.9 Example of a prediction that the size is too large

There is a trade-off between the accuracy and the size of the prediction, and the goal is to balance these two values. **Step 2**, where the zones are divided, is the step that has the most significant impact on this. In **Step 2a**, $k = 8$ is the optimal value choosing for K-Means Clustering. If k is too high, it means that there are many zones and each zone only contain a few locations, lowering the zones classification and grid positioning accuracy (but the size of prediction is tiny). On the contrary, if k is too small, the accuracy increases and so does the size of the prediction. The reason why the number of zones in **Step 2b** is 4 is the same as the reason for selecting k . Besides, the strategy for selecting one subset indicated in sub-step 4 in **Step 4** also plays an important role. If the strategy is choosing the subset that has the smallest intersection, the accuracy might drop a little, but the sizes of prediction are small. On the other hand, choosing the subset with the greatest intersection results in bigger predictions' size, which can slightly boost the accuracy. Two strategies are compared in **Table 5.4**, and it should be noted that the statistics in **Table 5.3** are generated using the strategy of choosing the smallest intersection.

Table 5.4 Comparison of two strategies

Strategy		Fold				
		1	2	3	4	5
Accuracy	Min	90.000%	82.051%	82.051%	92.307%	79.487%
	Max	90.000%	82.051%	84.615%	92.307%	82.051%
Mean size of all predictions	Min	1.800	1.820	1.769	1.692	1.384
	Max	1.925	1.846	1.923	1.769	1.512

Because this is a positioning algorithm, another statistic to examine is the distance error. As stated earlier, the dataset does not provide the measurements of the locations, so each location is considered a point in a XY-plane. The distance error is calculated as follows:

- If the prediction is a single-location prediction, the distance error is just the Euclidean distance between the prediction and the actual location.
- If the prediction is a multiple-location prediction, the centre point of all locations in the prediction is determined. Then the distance error is the Euclidean distance between that centre point and the actual location.

The root-mean-square error of five folds are listed in **Table 5.3**, and the values show a positive indicator since most of them are below 1 (except the 2nd fold). The fact that the root-mean-square error is minor implies that the incorrect predictions are not too far off the actual location. For instance, in line 9 in **Figure 5.8**, the incorrect prediction is (5, 3), and the actual location is (5, 4).

Finally, there are two main factors that decide the success of this grid positioning algorithm:

1. The accuracy of zones classification in Step 3: Although the accuracy achieved in **Table 5.2** is remarkable, it would be much better if it could reach almost 100% for all combinations of zones. Although neural networks have been attempted for this zones classification task, it has been found that the accuracy does not increase significantly and that the training time is expensive (it takes at least 2000 epochs to train).
2. The combination of zones from in Step 2: As previously stated, the zone labels should be distinct in order for the intersection to result in a small size. If the zone labels are too similar, the intersection will be a multiple-location prediction with very large size. And if we want the zone labels to be different, the combinations of zones when dividing in **Step 2** must be different:

- a. **Step 2a:** Since the combinations of zones in this step are divided based on TOA/RSS using K-Means Clustering, determining whether they are different is to use machine learning algorithms, which is what has been done in **Section 3**. Humans can perceive that the combinations of zones are different with their eyes (**Figure 5.3**), but an algorithm is needed to identify the differences for computers. In **Section 3**, the signals from four rooms are classified, and the result shows that they are highly distinct, which is why four combinations of zones in **Step 2a** are also distinct. Accordingly, it can be said that the task in **Section 3** is to verify the necessary condition for the grid positioning algorithm. Despite the fact that the findings from **Section 3** suggest that the signals from rooms with varying levels of clutter are distinct, it is necessary to verify them before inputting to this grid positioning algorithm.
- b. **Step 2b:** As the combinations of zones in this step are divided based on AOA, the room's position is the factor that determines the distinction between combinations of zones. To illustrate, the zones combination of the laboratory differs from the zones combination of the lobby because the positions of these two rooms are far from each other. On the contrary, the zones combinations of the laboratory and the corridor are almost identical since they are located close to one other

5.7 Compared with Existing Methods

There are a number of papers and articles that also present a grid-based positioning approach. The work from [5] proposed a system that utilises BLE devices to collect data assisting positioning. It was carried out in a room with 32 chairs, each one representing a grid point. The data gathered from the transmitters and receivers are then used to train the machine learning models. The results indicate that the algorithm Adaboost-SVM has the highest accuracy of 91.2% and its delay time is roughly 5 seconds. Another example is the work from [33], which performs positioning in a room divided into a 3*3 grid area (each grid location is 1.44 m²). This approach then uses the RSS of the signals to train the neural networks with varying layer sizes.

The size of the interior environment is obviously the most significant distinction between this thesis project's algorithm and the two from [5] and [33]. The dataset used in this thesis project is an entire floor of a building, whereas the other two works are only collected in just a single room. As a result, there are 196 grid points to locate in this thesis project, compared to 32 locations of [5] and 9 locations of [33]. Having stated that, this grid positioning algorithm is appropriate for determining which room of a floor an object or a person is in.

Nowadays, the most popular application of this is to determine the people in the same room for Covid-19 monitoring. When a person is tested positive for covid 19, the rooms they visit are logged, and people who shared those rooms with them are alerted.

Additionally, this grid positioning algorithm is incapable of performing multi-level environment placement. It is undeniable that multi-level environment placement is the hardest aspect of indoor positioning to achieve. [33] introduces a grid-based algorithm on a dataset gathered from four floors, and the accuracy is around 88%, which is promising. But in return, its setup is very complicated even though this is only four floors.

Chapter 6

6 Conclusion and Future Work

6.1 Conclusion

In conclusion, indoor positioning is undeniably a hot issue these days, with more and more research being conducted to advance the subject. Because there are several challenges in integrating indoor positioning, researchers are very interested in this field (there are a lot of solid solutions, but none of them is perfect in every scenario). For indoor positioning, a variety of technologies have been used, the most common of which is the radio-based technology family, with UWB being the most promising. In comparison with other technologies, UWB offers better accuracy, great bandwidth and power efficiency.

This thesis project focuses on designing a grid-based positioning algorithm on a floor of a building. The task of verifying the requisite condition for the algorithm is covered first in **Section 3**. Subsequently, the details of the grid positioning algorithms are explained in **Section 4**. The algorithm uses a variety of techniques, including signal processing and machine learning, to achieve remarkable accuracy (it varies from 80% to 90%). The accuracy can be further improved by enhancing the machine learning models when doing zones classification. Apart from accuracy, the prediction distance error is also examined, and it is not substantial, which is excellent. The success of the algorithm once again demonstrates that UWB is ideal for positioning, and the combination of three measures (TOA, RSS, and AOA) maximizes the UWB signal's utility.


One of the algorithm's key drawbacks is that it does not ensure single-location prediction. Consequently, there are still a few predictions that contain four or even five locations, and these predictions are meaningless, although they are correct. While setting up the experiment for gathering data, the rooms must be chosen such that their clutter levels are diverse, as well as their location. In other words, this problem can be solved by carefully selecting the rooms where the signal will be collected and verifying the signals before feeding them into the algorithm. Furthermore, while dividing the entire floor into zones, multiple experiments must be performed to determine the appropriate value for the hyperparameters so that the accuracy and size of the prediction are balanced. Ultimately, this positioning algorithm is suitable for detecting which room an object or a person is in. There are many applications, and the most prevalent is for Covid-19's spread tracking (determine who shared a room with a Covid-19 positive individual).

6.2 Future Work

The grid algorithm in this thesis project can be regarded as the offline phase of the *fingerprinting* process (fingerprinting is previously explained in **Section 2.2**). Therefore, the future plan is to conduct the online phase of it. In other words, the algorithm will need to be tested in a real-time scenario with participants carrying tags and wandering the floor. More problems (such as participants' walking pace) will arise by then, and the algorithm will undoubtedly need to be modified and improved. In addition, the algorithm should also be evaluated in a variety of interior settings (for example, in a level of a different building) to ensure that it does not simply work well in this situation.

7 Appendix

7.1 Exemption Approval



**THE UNIVERSITY
OF QUEENSLAND**
AUSTRALIA

6 November 2020

Dr Guohun Zhu
School of ITEE

Dear Dr Guohun Zhu,

Clearance Number: 2020002666
Project Title: *Classification of Indoor Environments/Localization Based on Graph Approaches using UWB Signals*

This project has been reviewed by the Office of Research Ethics and is deemed to be exempt from ethics review under the National Statement on Ethical Conduct in Human Research and University of Queensland policy.

1. Research that uses only existing collections of data that contain only non-identifiable data about human beings **AND** is of negligible risk, and is exempt from review ([National Statement §5.1.22](#)).
2. Audit or Quality Assurance activity


Chief Investigator	Dr Guohun Zhu
Associate Investigators	Prof Fangyan Dong (Ningbo University)
Data Source	It is available from IEEE data portal "2.4 ghz indoor channel measurements," 2018. [Online]. Available: http://dx.doi.org/10.21227/ggh1-6j32 It is also accessed from https://archive.ics.uci.edu/ml/datasets/2.4+GHZ+Indoor+Channel+Measurements
Completion Date	30 November 2021

Human Ethics Research Office
Cumbræ-Stewart Building #72
The University of Queensland
St Lucia, QLD 4072

CRICOS PROVIDER NUMBER 00025B

Please keep a copy of this document for your records.

Yours sincerely



Chris Rose Meyer
Senior Manager Human Research Ethics

Figure 7.1 Exemption approval

7.2 Dataset and Code

All the programming work in this thesis project is done using Google Colab, and source codes files can be found [here](#), while the dataset repository can be accessed [here](#).

7.3 Results

```

1. Location: (1, 2) - False Prediction: [(1, 4)]
2. Location: (6, 2) - Correct Prediction: [(5, 2), (5, 1), (5, 0), (6, 3), (6, 2)]
3. Location: (2, 13) - Correct Prediction: [(2, 13)]
4. Location: (12, 1) - Correct Prediction: [(12, 1), (12, 5), (12, 3), (12, 4), (12, 2)]
5. Location: (10, 10) - Correct Prediction: [(10, 10), (10, 9)]
6. Location: (5, 5) - Correct Prediction: [(5, 5)]
7. Location: (5, 0) - Correct Prediction: [(5, 2), (5, 1), (5, 0), (6, 3), (6, 2)]
8. Location: (9, 11) - Correct Prediction: [(9, 11)]
9. Location: (8, 10) - Correct Prediction: [(8, 9), (8, 10)]
10. Location: (9, 9) - Correct Prediction: [(9, 9)]
11. Location: (6, 8) - Correct Prediction: [(6, 8)]
12. Location: (0, 7) - False Prediction: [(1, 7)]
13. Location: (10, 6) - Correct Prediction: [(10, 6)]
14. Location: (10, 2) - Correct Prediction: [(10, 2), (10, 3)]
15. Location: (4, 8) - Correct Prediction: [(4, 8)]
16. Location: (12, 11) - Correct Prediction: [(12, 11), (12, 12)]
17. Location: (13, 2) - False Prediction: [(13, 1)]
18. Location: (0, 9) - Correct Prediction: [(1, 9), (2, 9), (0, 9)]
19. Location: (6, 5) - Correct Prediction: [(6, 5)]
20. Location: (6, 9) - Correct Prediction: [(9, 10), (6, 9), (7, 9)]
21. Location: (10, 5) - Correct Prediction: [(10, 5)]
22. Location: (2, 12) - Correct Prediction: [(4, 12), (2, 12), (3, 12)]
23. Location: (11, 1) - Correct Prediction: [(11, 1)]
24. Location: (8, 2) - Correct Prediction: [(8, 2)]
25. Location: (3, 4) - Correct Prediction: [(3, 4)]
26. Location: (6, 0) - Correct Prediction: [(6, 0)]
27. Location: (6, 3) - False Prediction: [(6, 1)]
28. Location: (6, 7) - False Prediction: [(5, 10)]
29. Location: (12, 6) - False Prediction: [(12, 0)]
30. Location: (7, 6) - Correct Prediction: [(7, 6), (6, 6)]
31. Location: (0, 13) - Correct Prediction: [(0, 13), (1, 13)]
32. Location: (1, 8) - Correct Prediction: [(1, 8)]
33. Location: (4, 4) - False Prediction: [(7, 2), (9, 2)]
34. Location: (10, 0) - Correct Prediction: [(10, 0)]
35. Location: (8, 4) - Correct Prediction: [(8, 4)]
36. Location: (12, 2) - Correct Prediction: [(12, 1), (12, 5), (12, 3), (12, 4), (12, 2)]
37. Location: (4, 12) - Correct Prediction: [(4, 12), (2, 12), (3, 12)]
38. Location: (9, 13) - Correct Prediction: [(8, 13), (9, 13)]
39. Location: (3, 1) - Correct Prediction: [(3, 1)]

```

Figure 7.2 The prediction of the algorithm for each location in the test (2^{nd} fold)

1.	Location: (11, 6)	- False Prediction:	[(11, 4)]
2.	Location: (10, 4)	- Correct Prediction:	[(10, 4)]
3.	Location: (11, 5)	- Correct Prediction:	[(11, 5)]
4.	Location: (12, 5)	- Correct Prediction:	[(12, 1), (12, 5), (12, 3), (12, 4), (12, 2)]
5.	Location: (6, 12)	- Correct Prediction:	[(6, 12)]
6.	Location: (13, 4)	- Correct Prediction:	[(13, 4)]
7.	Location: (6, 13)	- Correct Prediction:	[(6, 13), (7, 13)]
8.	Location: (5, 8)	- Correct Prediction:	[(5, 8)]
9.	Location: (4, 2)	- False Prediction:	[(4, 0)]
10.	Location: (12, 0)	- Correct Prediction:	[(12, 0)]
11.	Location: (1, 10)	- False Prediction:	[(2, 10)]
12.	Location: (7, 4)	- Correct Prediction:	[(7, 4)]
13.	Location: (5, 6)	- Correct Prediction:	[(5, 6)]
14.	Location: (13, 10)	- False Prediction:	[(11, 10)]
15.	Location: (7, 2)	- Correct Prediction:	[(7, 2)]
16.	Location: (1, 12)	- Correct Prediction:	[(1, 12), (0, 12)]
17.	Location: (4, 5)	- Correct Prediction:	[(4, 5)]
18.	Location: (4, 1)	- False Prediction:	[(4, 0)]
19.	Location: (4, 6)	- Correct Prediction:	[(4, 6)]
20.	Location: (7, 12)	- Correct Prediction:	[(7, 12), (9, 12), (6, 11), (7, 11)]
21.	Location: (5, 2)	- Correct Prediction:	[(5, 2), (5, 1), (5, 0), (6, 3), (6, 2)]
22.	Location: (11, 3)	- False Prediction:	[(11, 5)]
23.	Location: (13, 8)	- Correct Prediction:	[(13, 8), (13, 7)]
24.	Location: (13, 6)	- False Prediction:	[(12, 1), (12, 5), (12, 3), (12, 4), (12, 2)]
25.	Location: (8, 1)	- Correct Prediction:	[(8, 1), (8, 0)]
26.	Location: (0, 2)	- Correct Prediction:	[(0, 1), (1, 3), (0, 3), (1, 2), (0, 2)]
27.	Location: (3, 11)	- Correct Prediction:	[(3, 11)]
28.	Location: (2, 0)	- Correct Prediction:	[(2, 0)]
29.	Location: (7, 0)	- Correct Prediction:	[(7, 1), (7, 0)]
30.	Location: (2, 6)	- Correct Prediction:	[(2, 6)]
31.	Location: (8, 8)	- Correct Prediction:	[(8, 8)]
32.	Location: (0, 12)	- Correct Prediction:	[(1, 12), (0, 12)]
33.	Location: (11, 13)	- Correct Prediction:	[(11, 13)]
34.	Location: (3, 5)	- Correct Prediction:	[(3, 5)]
35.	Location: (9, 0)	- Correct Prediction:	[(9, 0)]
36.	Location: (5, 13)	- Correct Prediction:	[(5, 13)]
37.	Location: (6, 1)	- Correct Prediction:	[(6, 1)]
38.	Location: (0, 3)	- Correct Prediction:	[(0, 1), (1, 3), (0, 3), (1, 2), (0, 2)]
39.	Location: (6, 10)	- Correct Prediction:	[(6, 10), (7, 10)]

Figure 7.3 The prediction of the algorithm for each location in the test set (3^d fold)

1.	Location: (2, 1)	- Correct Prediction: [(3, 2), (2, 1)]
2.	Location: (2, 7)	- Correct Prediction: [(2, 7)]
3.	Location: (5, 3)	- False Prediction: [(4, 0)]
4.	Location: (12, 3)	- Correct Prediction: [(12, 1), (12, 5), (12, 3), (12, 4), (12, 2)]
5.	Location: (7, 9)	- Correct Prediction: [(9, 10), (6, 9), (7, 9)]
6.	Location: (12, 4)	- Correct Prediction: [(12, 1), (12, 5), (12, 3), (12, 4), (12, 2)]
7.	Location: (12, 9)	- Correct Prediction: [(12, 9)]
8.	Location: (10, 11)	- Correct Prediction: [(10, 11)]
9.	Location: (4, 11)	- Correct Prediction: [(4, 10), (4, 11)]
10.	Location: (7, 7)	- Correct Prediction: [(7, 7)]
11.	Location: (8, 7)	- Correct Prediction: [(8, 7)]
12.	Location: (4, 3)	- Correct Prediction: [(4, 3)]
13.	Location: (5, 12)	- Correct Prediction: [(5, 12)]
14.	Location: (0, 4)	- Correct Prediction: [(0, 4)]
15.	Location: (1, 7)	- Correct Prediction: [(1, 7)]
16.	Location: (2, 8)	- Correct Prediction: [(2, 8)]
17.	Location: (3, 3)	- Correct Prediction: [(3, 3)]
18.	Location: (7, 13)	- Correct Prediction: [(6, 13), (7, 13)]
19.	Location: (4, 0)	- Correct Prediction: [(4, 0)]
20.	Location: (13, 12)	- Correct Prediction: [(13, 12), (13, 11)]
21.	Location: (10, 7)	- Correct Prediction: [(10, 7)]
22.	Location: (9, 10)	- Correct Prediction: [(9, 10), (6, 9), (7, 9)]
23.	Location: (7, 5)	- Correct Prediction: [(7, 5)]
24.	Location: (3, 13)	- Correct Prediction: [(4, 13), (3, 13)]
25.	Location: (9, 12)	- Correct Prediction: [(7, 12), (9, 12), (6, 11), (7, 11)]
26.	Location: (9, 2)	- Correct Prediction: [(9, 2)]
27.	Location: (1, 6)	- Correct Prediction: [(1, 6)]
28.	Location: (13, 7)	- Correct Prediction: [(13, 8), (13, 7)]
29.	Location: (0, 11)	- False Prediction: [(0, 13), (1, 13)]
30.	Location: (9, 6)	- Correct Prediction: [(9, 6)]
31.	Location: (9, 8)	- Correct Prediction: [(9, 8)]
32.	Location: (3, 10)	- Correct Prediction: [(3, 10)]
33.	Location: (8, 13)	- Correct Prediction: [(8, 13), (9, 13)]
34.	Location: (5, 1)	- False Prediction: [(6, 1)]
35.	Location: (13, 0)	- Correct Prediction: [(13, 0)]
36.	Location: (4, 9)	- Correct Prediction: [(3, 9), (4, 9)]
37.	Location: (10, 9)	- Correct Prediction: [(10, 10), (10, 9)]
38.	Location: (2, 2)	- Correct Prediction: [(2, 2), (2, 3)]
39.	Location: (12, 12)	- Correct Prediction: [(12, 11), (12, 12)]

Figure 7.4 The prediction of the algorithm for each location in the test set (4th fold)

1.	Location:	(6, 11)	- Correct Prediction:	[(7, 12), (9, 12), (6, 11), (7, 11)]
2.	Location:	(13, 13)	- Correct Prediction:	[(13, 13)]
3.	Location:	(2, 10)	- Correct Prediction:	[(2, 10)]
4.	Location:	(3, 7)	- Correct Prediction:	[(4, 7), (3, 7)]
5.	Location:	(5, 9)	- Correct Prediction:	[(5, 9)]
6.	Location:	(13, 1)	- Correct Prediction:	[(13, 1)]
7.	Location:	(1, 1)	- Correct Prediction:	[(1, 1), (1, 0)]
8.	Location:	(10, 12)	- Correct Prediction:	[(10, 12)]
9.	Location:	(3, 8)	- Correct Prediction:	[(3, 8)]
10.	Location:	(2, 9)	- Correct Prediction:	[(1, 9), (2, 9), (0, 9)]
11.	Location:	(8, 5)	- False Prediction:	[(6, 5)]
12.	Location:	(1, 0)	- Correct Prediction:	[(1, 1), (1, 0)]
13.	Location:	(3, 0)	- Correct Prediction:	[(3, 0)]
14.	Location:	(1, 11)	- Correct Prediction:	[(1, 10), (1, 11)]
15.	Location:	(5, 7)	- Correct Prediction:	[(5, 7)]
16.	Location:	(10, 1)	- Correct Prediction:	[(10, 1)]
17.	Location:	(9, 5)	- Correct Prediction:	[(9, 5)]
18.	Location:	(2, 3)	- Correct Prediction:	[(2, 2), (2, 3)]
19.	Location:	(0, 6)	- Correct Prediction:	[(0, 6)]
20.	Location:	(12, 13)	- Correct Prediction:	[(12, 13)]
21.	Location:	(11, 12)	- Correct Prediction:	[(11, 12)]
22.	Location:	(1, 4)	- Correct Prediction:	[(1, 4)]
23.	Location:	(9, 7)	- False Prediction:	[(8, 7)]
24.	Location:	(8, 3)	- False Prediction:	[(9, 1)]
25.	Location:	(5, 10)	- False Prediction:	[(5, 11)]
26.	Location:	(6, 6)	- False Prediction:	[(8, 7), (6, 8)]
27.	Location:	(2, 5)	- Correct Prediction:	[(2, 5)]
28.	Location:	(11, 0)	- Correct Prediction:	[(11, 0)]
29.	Location:	(12, 8)	- False Prediction:	[(11, 7)]
30.	Location:	(3, 12)	- Correct Prediction:	[(4, 12), (2, 12), (3, 12)]
31.	Location:	(12, 10)	- Correct Prediction:	[(12, 10), (11, 9)]
32.	Location:	(4, 7)	- Correct Prediction:	[(4, 7), (3, 7)]
33.	Location:	(7, 8)	- Correct Prediction:	[(7, 8)]
34.	Location:	(2, 11)	- Correct Prediction:	[(2, 11)]
35.	Location:	(11, 10)	- Correct Prediction:	[(11, 10)]
36.	Location:	(4, 10)	- False Prediction:	[(3, 11)]
37.	Location:	(11, 11)	- Correct Prediction:	[(11, 11)]
38.	Location:	(13, 9)	- Correct Prediction:	[(13, 9)]
39.	Location:	(11, 2)	- False Prediction:	[(11, 5)]

Figure 7.5 The prediction of the algorithm for each location in the test set (5th fold)

8 Bibliography

- [1] A. Hameed and H. A. Ahmed, "2018 12th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)," ed: IEEE, 2018, pp. 1-5.
- [2] Sewio. "Indoor Location Technologies." <https://www.sewio.net/indoor-location-tracking-and-positioning/> (accessed September 12, 2021).
- [3] Apple. "iBeacon." <https://developer.apple.com/ibeacon/> (accessed September 12,, 2021).
- [4] D. Oosterlinck, D. F. Benoit, P. Baecke, and N. Van de Weghe, "Bluetooth tracking of humans in an indoor environment: An application to shopping mall visits," *Applied geography (Sevenoaks)*, vol. 78, pp. 55-65, 2017, doi: 10.1016/j.apgeog.2016.11.005.
- [5] Q. Hu, J. Yang, P. Qin, S. Fong, and J. Guo, "Could or could not of Grid-Loc: grid BLE structure for indoor localisation system using machine learning," *Service oriented computing and applications*, vol. 14, no. 3, pp. 161-174, 2020, doi: 10.1007/s11761-020-00292-z.
- [6] M. Ghavami, L. B. Michael, R. Kohno, and R. Kohno, *Ultra wideband signals and systems in communication engineering*, 2nd ed. ed. Chichester, England Hoboken, NJ: John Wiley, 2007.
- [7] C. Xinyun, W. Ge, L. Jinqiao, X. Junye, and L. Qi, "Review on Ultra Wide Band Indoor Localization," *Indonesian Journal of Computing, Engineering and Design*, vol. 2, no. 2, pp. 99-110, 2020, doi: 10.35806/ijoced.v2i2.118.
- [8] M. Truijens, X. Wang, H. d. Graaf, and J. J. Liu, "Evaluating the Performance of Absolute RSSI Positioning Algorithm-Based Microzoning and RFID in Construction Materials Tracking," *Mathematical problems in engineering*, vol. 2014, 2014, doi: 10.1155/2014/784395.
- [9] S. Sand, A. Dammann, and C. Mensing, *Positioning in wireless communications systems*. Hoboken Chichester, England: Wiley, 2014.
- [10] L. Hui, H. Darabi, P. Banerjee, and L. Jing, "Survey of Wireless Indoor Positioning Techniques and Systems," *IEEE transactions on systems, man and cybernetics. Part C, Applications and reviews*, vol. 37, no. 6, pp. 1067-1080, 2007, doi: 10.1109/TSMCC.2007.905750.
- [11] Z. Yang, Z. Zhou, and Y. Liu, "From RSSI to CSI: Indoor localization via channel response," *ACM computing surveys*, vol. 46, no. 2, pp. 1-32, 2013, doi: 10.1145/2543581.2543592.
- [12] T. L. N. Nguyen and Y. Shin, "A new approach for positioning based on AOA measurements," in *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, 2013: IEEE, pp. 208-211.
- [13] C. Zhou, J. Liu, M. Sheng, Y. Zheng, and J. Li, "Exploiting Fingerprint Correlation for Fingerprint-Based Indoor Localization: A Deep Learning Based Approach," *IEEE transactions on vehicular technology*, vol. 70, no. 6, pp. 5762-5774, 2021, doi: 10.1109/TVT.2021.3075539.

BIBLIOGRAPHY

- [14] Q. Yihong and H. Kobayashi, "GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489)," vol. 7, ed: IEEE, 2003, pp. 4079-4083 vol.7.
- [15] K. Technologies, "S-Parameter Measurements: Basics for High Speed Digital Engineers." [Online]. Available: <https://literature.cdn.keysight.com/litweb/pdf/5991-3736ENDL.pdf>
- [16] B. Boashash, *Time frequency signal analysis and processing : a comprehensive reference*, Second edition. ed. Amsterdam & Boston: Academic Press, 2016.
- [17] I. Oppermann, M. Hämmäläinen, and J. Iinatti, *UWB theory and applications*. Chichester: John Wiley, 2004.
- [18] M. I. AlHajri, N. T. Ali, and R. M. Shubair, "Classification of Indoor Environments for IoT Applications: A Machine Learning Approach," *IEEE antennas and wireless propagation letters*, vol. 17, no. 12, pp. 2164-2168, 2018, doi: 10.1109/LAWP.2018.2869548.
- [19] M. I. AlHajri, N. Alsindi, N. T. Ali, and R. M. Shubair, "2016 IEEE International Symposium on Antennas and Propagation (APSURSI)," ed: IEEE, 2016, pp. 1447-1448.
- [20] E. Salahat, A. Kulaib, N. Ali, and R. Shubair, "Exploring Symmetry in Wireless Propagation Channels," 2017.
- [21] G. Zhu, F. Dong, and N. Pang, "2020 IEEE Symposium Series on Computational Intelligence (SSCI)," ed: IEEE, 2020, pp. 197-201.
- [22] A. Taner and H. Mohammed Muwafaq Noori, "A Clustering-Based Approach for Improving the Accuracy of UWB Sensor-Based Indoor Positioning System," *Mobile information systems*, vol. 2019, 2019, doi: 10.1155/2019/6372073.
- [23] P. Kukolev, A. Chandra, T. Mikulášek, and A. Prokeš, "Out-of-vehicle time-of-arrival-based localization in ultra-wide band," *International journal of distributed sensor networks*, vol. 12, no. 8, p. 155014771666552, 2016, doi: 10.1177/1550147716665522.
- [24] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," 2018.
- [25] A. P. Andy Coenen. "Understanding UMAP." <https://pair-code.github.io/understanding-umap/> (accessed August 29, 2021).
- [26] E. Alpaydin, *Introduction to machine learning*, Fourth edition. ed. Cambridge, Massachusetts: The MIT Press, 2020.
- [27] Scikit-learn. "Support Vector Machines." <https://scikit-learn.org/stable/modules/svm.html> (accessed August 29, 2021).
- [28] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of Statistical Learning*. New York: Springer, 2009.
- [29] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of machine learning research*, 2011, doi: 10.5555/1953048.2078195.
- [30] J. Brownlee, "LSTMs for Human Activity Recognition Time Series Classification," vol. 2021, ed: Machine Learning Mastery, 2018.
- [31] Keras. (2015). GitHub. [Online]. Available: <https://keras.io>

BIBLIOGRAPHY

- [32] TutorialsPoint. "Analyzing Signals."
https://www.tutorialspoint.com/principles_of_communication/principles_of_communication_analyzing_signals.htm (accessed September 5, 2021).
- [33] N. G. Jeevarathnam, "Grid-Based RFID Indoor Localization Using Tag Read Count and Received Signal Strength Measurements," ed: ProQuest Dissertations Publishing, 2017.