FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Normalization and smoothing of RSSI values of Bluetooth connection
Student:	Filip Špaček
Supervisor:	Ing. Daniel Vašata, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2023/2024

Instructions

For Bluetooth devices, one of the important parameters of the signal is its actual strength, measured by the Received Signal Strength Indicator (RSSI). Changes in its intensity can be used to detect when two Bluetooth devices are approaching or moving away.

The aim of this thesis is to investigate the data provided by 2N TELEKOMUNIKACE a.s., which contains information about the RSSI strength between mobile phones and remote unlocking door units. It turns out that for different types of phones, the RSSI values are different. Furthermore, the RSSI values depend on the type of packets (advertising or data packets). To successfully use algorithms that can predict the proximity of a phone to a unit based on the RSSI strength, it is helpful to be able to normalize the data. In addition to normalization, the goal of this thesis is to try to smooth out the noisy RSSI values and investigate how this affects the success of predicting the phone's proximity to the unit.

The specific assignment guidelines are:
1. Get familiar with the data provided by 2N TELEKOMUNIKACE a.s. from different scenarios of mobile phones approaching and moving away from unlocking door units. Describe the data in the thesis.
2. Analyze and discuss the differences in RSSI values depending on the type of packets. Design and implement a suitable normalization method that unifies the RSSI values for both types.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 10 February 2023 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

3. Analyze and discuss the differences in RSSI values between different brands/models of mobile phones. Attempt to design and implement a method by which RSSI values could be unified between brands/models.
4. Research suitable time series smoothing methods. Select at least two of them and apply them to smoothen RSSI values.
5. For items 2-4 above, use the provided data to evaluate the selected approaches and test the impact on the performance of the algorithms that trigger remote unlocking of the door units based on the phone's proximity.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 10 February 2023 in Prague.

Bachelor's thesis

NORMALIZATION AND SMOOTHING OF RSSI VALUES OF BLUETOOTH CONNECTION

Filip Špaček

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Ing. Daniel Vašata, Ph.D.
May 11, 2023

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis: Špaček Filip. *Normalization and smoothing of RSSI values of Bluetooth connection.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Contents

List of Figures

List of Tables

I want to thank my thesis supervisor, Ing. Daniel Vašata Ph.D., for his exceptional guidance and support. His input was invaluable and he also pushed me to give my best. I'm also grateful for my family and friends' unwavering support throughout this journey. Thank you for making this all possible.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. I further declare that I have concluded an agreement with the Czech Technical University in Prague, on the basis of which the Czech Technical University in Prague has waived the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant to Section 60(1) of the Copyright Act. This fact does not affect the provisions of Section 47b of the Act No. 111/1998 Coll., on Higher Education Act, as amended.

In Prague on May 11, 2023 .....

Abstract

This thesis explores the effects of RSSI time series normalization and smoothing. It implements several different methods such as exponential smoothing, moving average smoothing, and Savitzky-Golay smoothing. It also proposes a normalization technique for compensating differences between RSSI values of distinct packet types. Proposed methods were tested on the existing approach detection model and results were compared.

Keywords approach detection, BLE, data normalization, RSSI, time series smoothing

Abstrakt

Tato práce zkoumá účinky normalizace a vyhlazování časových řad RSSI. Implementuje několik různých metod, jako je exponenciální vyhlazování, klouzavý průměr a Savitzky-Golayův algoritmus. Také navrhuje normalizační techniku pro kompenzaci rozdílů mezi hodnotami RSSI pro různé typů paketů. Navržené metody byly testovány na existujícím modelu detekce přístupu a výsledky byly porovnány.

Klíčová slova detekce příchodu, BLE, normalizace dat, RSSI, vyhlazování časových řad

Abbreviations

BLE	Bluetooth Low Energy
GAP	Generic access profile
RF	Radio frequency
RSSI	Received Signal Strength Indicator
RX	Absolute value of received signal
SMA	Simple moving average

Introduction

On August 16th, 1994 IBM released its Simon personal communicator, which is nowadays known as the first-ever smartphone. It was able to make telephone calls, it additionally had 11 built-in programs: a to-do list, calendar, calculator, appointment scheduler, electronic sketch pad, world time clock, input screen keyboards, handwritten annotations, and address book. Not so impressive compared to the countless functionalities that modern smartphones possess. But it paved the way for further smartphone development and smartphones despite their humble beginnings became an everyday necessity for the masses of people all around the globe.

To take the most out of this further development has to be made, improving not only smartphones themselves but also integrating them with other devices. This is happening in all sorts of fields, a smartphone can be connected to a car, through an infotainment system enabling users to use plenty of their smartphone features, like navigation, phone calls, or audio streaming all while driving a car. At home, people can connect to the television and stream photos or music through the handy device, that they are carrying in their pockets. All of this is possible due to the existence of Bluetooth, a wireless communication standard.

Bluetooth is definitely one of the reasons smartphones are such versatile devices. A few decades ago, if you wanted to play music, you needed a cassette player or a record player. If you wanted to record and share videos with your friends and family, you needed a camcorder. And in addition to these devices, you also needed the right cables to connect everything together. Nowadays, a smartphone can handle all of this, and thanks to Bluetooth you do not have to worry about the cables.

People have quickly become used to controlling many things through their smartphones. For this cause, 2N TELEKOMUNIKACE a.s developed WaveKey technology that enables it to satisfy the increasing demand for remote access technology. One fundamental part of the WaveKey technology is a machine learning algorithm that triggers remote unlocking of the door unit based on smartphone proximity. This thesis aims to explore ways of improving the existing model focusing more on the preprocessing, so the model can make better predictions.

Aim of the thesis

The overall goal of this thesis is to analyze data provided by 2N TELEKOMUNIKACE a.s, from different scenarios of smartphones moving towards or away from door unlocking units, and then research and implement suitable preprocessing techniques to improve the performance of the algorithm that triggers remote door unlocking. The individual goals are summarized in the list below:

- Researching and implementing suitable time series smoothing methods.
- Exploring data from different testing scenarios
- Analyzing the differences of RSSI depending on packet type, and for various brands and models. Discussing the possible normalization of RSSI.
- Testing the effect of proposed preprocessing techniques on model predictions.

Chapter 1

Bluetooth and RSSI

According to one of the founders of Bluetooth technology Jim Kardach Bluetooth got its name from King Harald Bluetooth, because he was known for uniting Scandinavia just as this new technology was intended to unite PC and cellular industries with a short-range wireless link. Even though Bluetooth was only indented to be a placeholder until some better name will be figured out, it prevailed and it is used to this day. For the whole origin story see [1].

And it truly honors its name since nowadays Bluetooth is embedded in virtually all mobile devices and also headphones, speakers, in-car entertainment systems, etc. It evolved into a powerful industrial and domestic IoT (Internet of Things) connectivity solution. And because it is so widespread, it makes for a great choice as technology for indoor positioning and approach detection.

That is why this chapter will introduce the very basics of this short-range wireless communication technology standard.

1.1 Technology overview

As mentioned above one of the main advantages of Bluetooth technology is its incredible flexibility. Bluetooth offers two radio options so let's take a quick look at these options.

1.1.1 Bluetooth Classic

As mentioned in [2] the Bluetooth Classic radio, also known as Bluetooth Basic, is a low-power radio capable of streaming data over 79 channels in the 2.4GHz unlicensed ISM[1] frequency band. Supporting point-to-point connection, the Bluetooth Classic is mainly used for wireless audio streaming. It also enables data transfer applications and mobile printing.

1.1.2 Bluetooth Low Energy

As mentioned in [2] the Bluetooth Low Energy radio is designed for very low-power operation. It is capable of transmitting data over 40 channels in the 2.4GHz unlicensed ISM frequency band (same as the Bluetooth Classic). Bluetooth LE (BLE) is super flexible so it enables developers to create products that meet the unique connectivity requirements of the market. In contrast with Bluetooth Low Energy, BLE supports multiple communication topologies (see figure 1.1), expanding from only the point-to-point approach of the Bluetooth Classic, to broadcast, and most

[1] Refers to radio spectrum reserved for industrial, scientific, and medical purposes

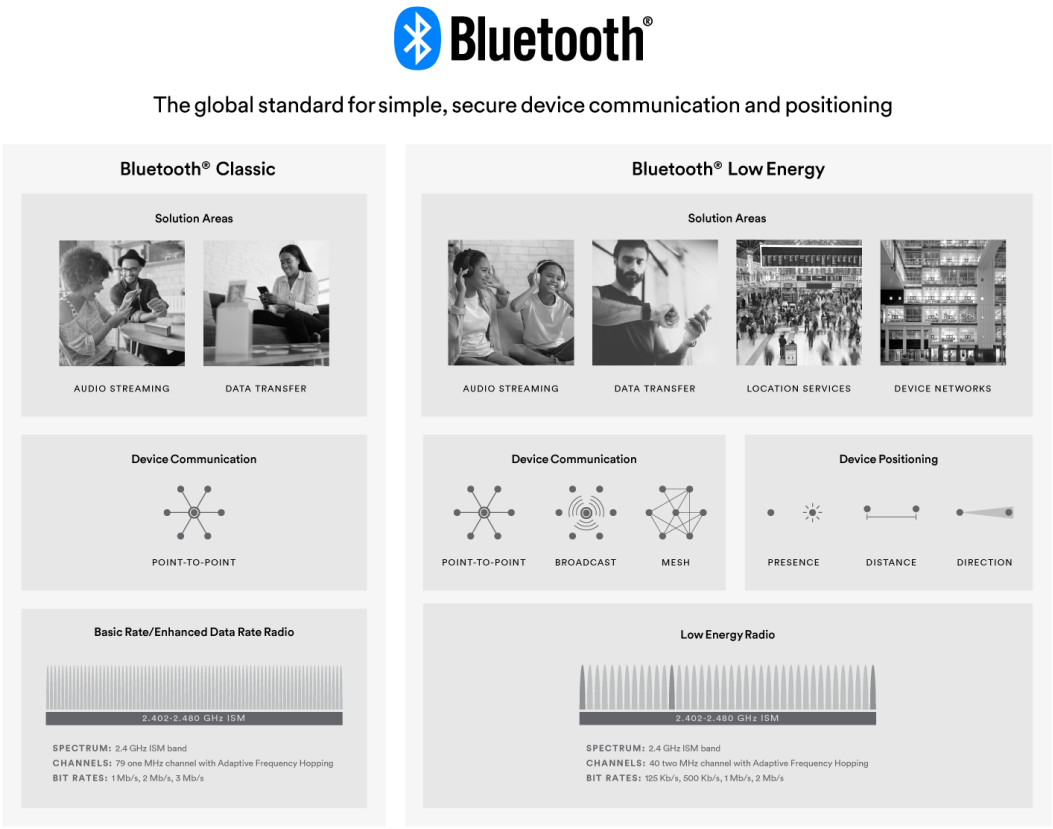


Figure 1.1 Bluetooth technologies comparison. Figure taken from [3].

recently, mesh, enabling the Bluetooth technology to create reliable, large-scale device networks. Currently, the BLE is widely used as a device positioning technology to address the increasing demand for accurate indoor localization. BLE now includes features that help to determine the presence, distance, or even direction of another device.

1.2 BLE protocol stack

This section will further explore BLE technology since it is the one used in positioning applications. The hierarchy of individual layers can be seen in figure 1.2.

1.2.1 Generic access profile

In [4] is explained a very important protocol of the BLE stack called Generic access profile. It handles the broadcasting of advertising packets as well as connection (in cooperation with the Security Manager layer). As mentioned above, BLE recognizes 40 different radio frequency channels. Their frequency range from 2402 MHz to 2480 MHz, which means that each channel center is separated by 2 MHz. There are 3 main advertising channels as shown in figure 1.1. Other 37 are used for data transfer during connection or for secondary advertisement. GAP recognizes these 4 roles:

- **Broadcaster** sends advertisements and does not support connection with other devices.

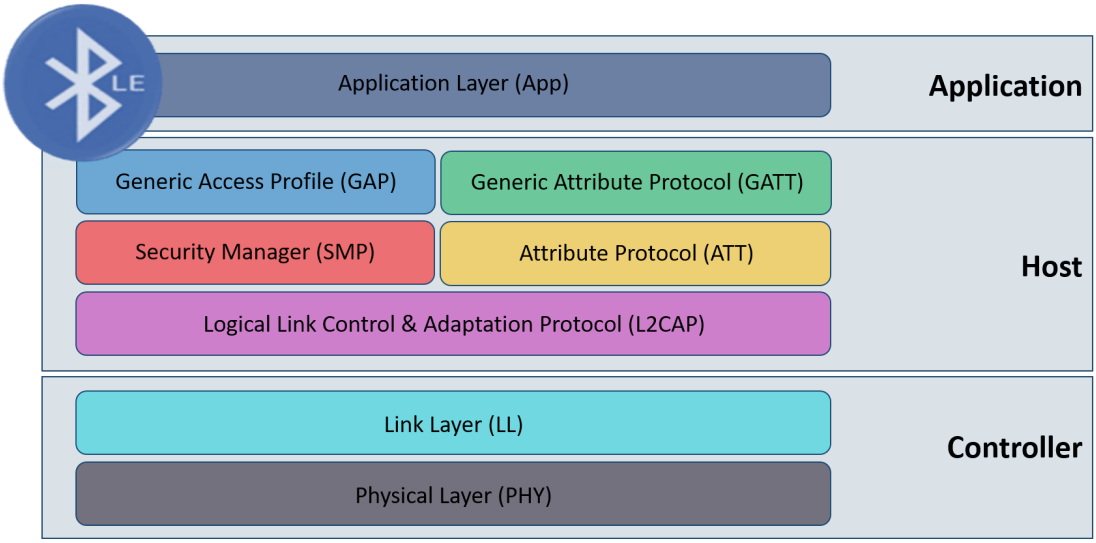


Figure 1.2 BLE stack overview. Figure taken from [5].

- **Observer** observes for advertisements from other devices but does not initiate the connection.
- **Peripheral** sends advertisements and also allows connection with other devices.
- **Central** does not send any advertisements, it only scans for the other devices, and in case it founds a peripheral device it can initiate a connection.

Advertisement

As explained in [6] BLE device can operate in one of two modes. Advertisements are packets sent by a device that wants to be discovered. Advertisements are sent in an interval that ranges from 20 ms to 10.24 s and increments by 625 μs. Some advertisements are so-called scannable which means that scanning BLE devices can send requests for additional data to the advertiser.

Connection

The second mode the BLE device can operate in is the connection mode, see [7] for more details. This mode allows for persistent, synchronized data exchange between two BLE devices. This is different from the advertisement mode which sends data only one way. After the connection is established the Central BLE device becomes a master and the Peripheral becomes a slave to the Central. The master device has overall control over connection parameters. Some of the most important ones are:

- **Connection interval** defines when the connection event occurs. Connection interval lasts from 7.5 ms to 4 s and increments of 1.25 ms. Each connection event is started by the master sending a packet to the slave, the slave then has to respond with a data packet.
- **Slave latency** defines a number of connection events that the slave can skip if it does not have any data to send to the master. The maximum value of slave latency is 500.
- **Supervision timeout** defines a time in which the connection is considered lost. It ranges from 100 ms to 32 sec and increments by 10 ms.

- **Channel Hop** defines which channel will be used next for a connection event. Since each connection event occurs on a different radio frequency channel a technique called channel hopping is used to change the channel after each connection event. If the selected channel is not included in the channel map (list of channels used for communication, defined by the master) then it is mapped to a different channel using a specific formula.

1.3 RSSI

The RSSI is a relative measurement of received signal strength. It is basically the signal strength percentage. This means that the higher the RSSI value, the stronger the received signal. There is no standardized version of RSSI calculation since it is not bound to any physical parameter. Due to this a chip from company A might have RSSI values from 0 to 100, but a company B chip will have RSSI from 0 to 127. The overall allowable range is 0-255 as mentioned in [7]. These differences have to be taken into account whenever RSSI is used.

The RSSI is similar to the RX (received input), which is a measurement of the signal power level received by an antenna. RX is measured in milliWatts (mW) or decibel-milliwatts (dBm). Decibel-milliwatt is a logarithmic unit of decibels, referenced to 1 milliwatt(see [8] for more information). Some specific chips use a more standardized version of RSSI and have mappings to a particular physical RX value as mentioned in [9]. And since only the RSSI is available in high-level API most of the applications measuring signal strength use RSSI.

Since the dBm has negative values then measurements close to 0 are perfect (Unachievable in real-life scenarios). But the overall rule is that a higher value means a better signal.

1.4 Influence of external factors to RSSI

"The received signal strength indicator (RSSI), a common metric in most networking hardware, has been reputed as a very unreliable method for doing the job, due to its vulnerability to environmental factors. Nevertheless, it still remains the most prevalent estimator of the distance between agents on many research projects." [10, p. 1537]

This quote sums it up pretty well. The RSSI has proven itself to be a very unreliable metric for distance estimations. But since developers usually do not have any better option it is used in many localization systems. This unreliability is due to noisiness which is caused by numerous factors, most of which are dependent on the current environment, used device, or setup. So let's take a deeper look at how some of them affect RSSI measurements.

1.4.1 Temperature

In [11] the authors investigated the relationship between measured RSSI and temperature. The results showed a linear relationship and strong negative correlation between temperature and RSSI, for devices further from each other than 5 m. For closer distances, the correlation was weaker as shown in table 1.1.

Table 1.1 Correlation coefficients between temperature and RSSI.

Corr(T,RSSI)		
Samples	Distance $\geq$ 5m	Distance = 3m
Anchor 1	-0.69	-0.37
Anchor 2	-0.73	-0.20
Anchor 3	-0.92	-0.75

The measurements were taken indoors and the temperature fluctuated between 22°C and 25.5°C. According to these findings, even small temperature variations have an impact on measured RSSI. For instance, when the temperature rose from 23°C to 25°C, the RSSI collected by 7m distant Anchor 2 decreased by 8dBm.

1.4.2 Relative humidity

In [11] the authors continued with exploring the effects of relative humidity on RSSI. Measurements were taken indoors and researchers selected measurements where the temperature was 23°C. The data showed a linear relationship between relative humidity and measured RSSI, with a positive correlation as shown in table 1.2. The difference in correlation coefficients for each anchor, researchers attribute to different characteristics of each anchor's location inside the room.

Table 1.2 Correlation coefficients between relative humidity and RSSI.

Corr(RH,RSSI)		
Samples	Distance $\geq$ 5m	Distance = 3m
Anchor 1	0.80	0.21
Anchor 2	0.84	0.67
Anchor 3	0.71	0.17

In data collected by 5m distant anchor 1, relative humidity varied between 38% and 47%, and with constant temperature, 23°C, the RSSI increased from -77dBm to -72dBm. But for a distance of 3m, with the same anchor, the RSSI is almost unchanged.

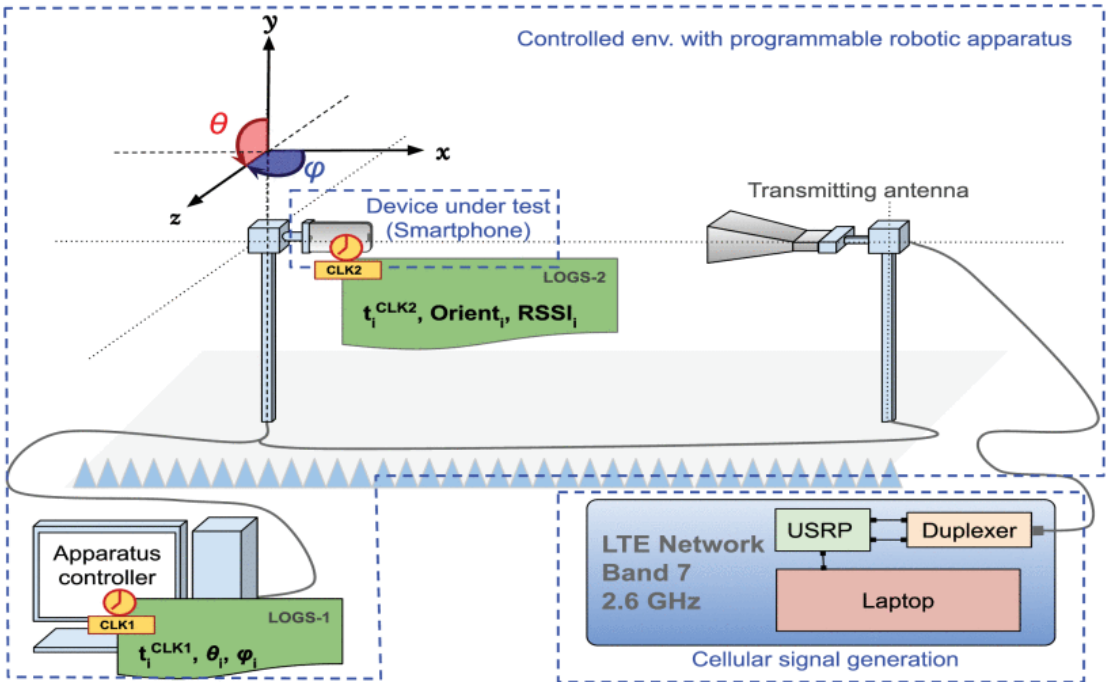


Figure 1.3 Setup used for measuring effects of rotation on RSSI. Note that the setup with LTE source but the setup with BLE source was made analogically as explained in [12].

1.4.3 Orientation

In paper [12] the authors explain the relationship between a smartphone's position and measured RSSI. Researchers took measurements in a controlled environment shown in figure 1.3. They used an anechoic chamber with programmable robotic equipment for both the transmission and reception sides. They created a two-axis positioning system that rotates along the two axes x and y:

- ϕ: azimuth, the angle between x and z axes,
- θ: roll, the angle between y and z axes,

y can rotate 180°. (from −90° to 90°), whereas x can make a 360° rotation. This allows for two-axis rotation, which leads to obtaining RSSI measurements using smartphones in different orientations. The reception and transmission are separated by 4 m. An Arduino dongle was used as the source, running Bluetooth 4.0 Low Energy. The dongle generates the Bluetooth signal and transmits it through its embedded antennas. A Nexus 5× smartphone running Android 7 was used for measuring received RSSI.

They tried rotating the smartphone into 40 different positions in the above-mentioned experimental setup and took measurements. Figure 1.4 shows measured RSSI in all of the positions. It clearly shows that BLE RSSI is very sensitive to the device rotation with a maximum of 23dBm difference between the minimum of -76dBm and the maximum of -53dBm measured RSSI.

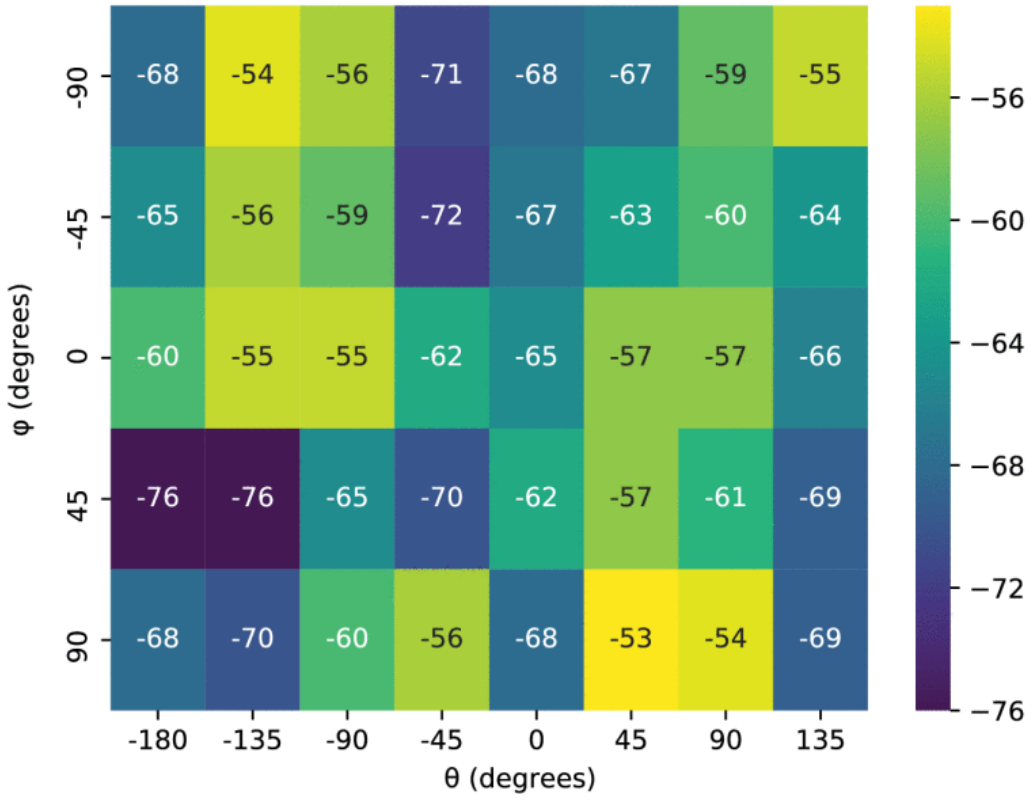


Figure 1.4 Measured RSSI for 40 different smartphone orientations. Figure taken from [12].

This proves that also the orientation of the receiving device affects RSSI measurements. And it occurs because the rotation changes the polarization of the antennas. According to antenna theory ideal, called polarization matching, means that the receiver and the transmitter have the

same polarization, which leads to minimal power loss. On the other hand, cross-polarization yields minimal power. Since most devices typically use only 1 antenna for BLE communication, and this antenna is linearly polarized, the power loss due to polarization between transmitting and receiving antennas, where ψ is the angle of rotation between these two antennas, can be described by the Polarization Loss Factor (PLF) [13]. The formula for this:

$$\mathrm{PLF} = \cos^2(\psi). \tag{1.1}$$

Another factor that causes RSSI to be so sensitive to positioning is the inhomogeneous character of the smartphone's construction which causes RF signal to propagate more easily through some parts of the devices than others. Is also safe to assume that distinct smartphone models will react differently to the rotation, due to inconsistencies in the location of the antenna used for Bluetooth communication and also due to the fact that manufacturers use various types of materials and designs to make their products.

Finally, researchers performed measurements outside of the anechoic chamber, in a realistic environment, an office. Figure 1.5 shows the measured BLE RSSI and its variability with the smartphone orientation. It is obvious that orientation has, yet again large impact on measured RSSI for both BLE sources. What's more, the minimum RSSI was not measured for the same orientation. This is due to the different polarization and construction of the two source devices.

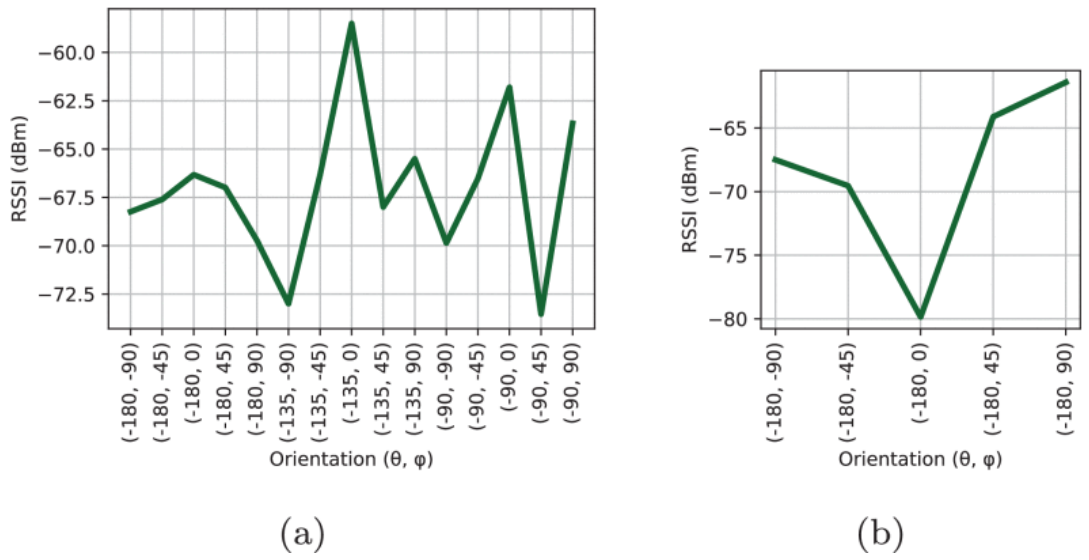


Figure 1.5 Measured RSSI for different smartphone orientations. On the left-hand side (a), the source is Arduino Dongle. On the right-hand side (b), the source is a smartwatch. Figure taken from [12].

1.4.4 BLE channels

In [14] the authors explored the effects of different changing advertisement channels on measured RSSI. As explained in section 1.1.2 BLE uses 40 different RF channels to transfer information between devices. Three of those channels (37, 38, and 39) are reserved for advertising. The center frequencies are, 2.402GHz, 2.426GHz, and 2.480GHz respectively. Since almost all devices have a frequency-dependent gain of transmitter G_t and gain of receiver G_r, the measured RSSI is affected by the selection of an RF channel. If transmitted power is P_t, then the power of received

wireless signal P_r traveling along distance d in unobstructed space, where λ is the wavelength of the signal, is explained by the equation:

$$P_r = P_t \cdot G_t \cdot G_r \cdot (\frac{\lambda}{4\pi d})^2. \tag{1.2}$$

Since $\lambda = \frac{c}{f}$, where c is the speed of light and f is the frequency, then the overall received power depends also on the frequency, the higher the f lesser P_r, so the channel selection itself directly affects P_r as explained in [14].

Packets sent on different channels also propagate differently in a real environment. Signals can get obstructed, reflected, or diffracted by objects in the environment. This means that multiple replicas of the same signal reach the receiving antenna. These replicas can make the signal stronger, or weaker depending on the type of interference. Signals of various frequencies from different channels react differently to this distortion influencing measured RSSI in unique ways as mentioned in [14]. Figure 1.6 shows the effects of different BLE advertising channels on RSSI.

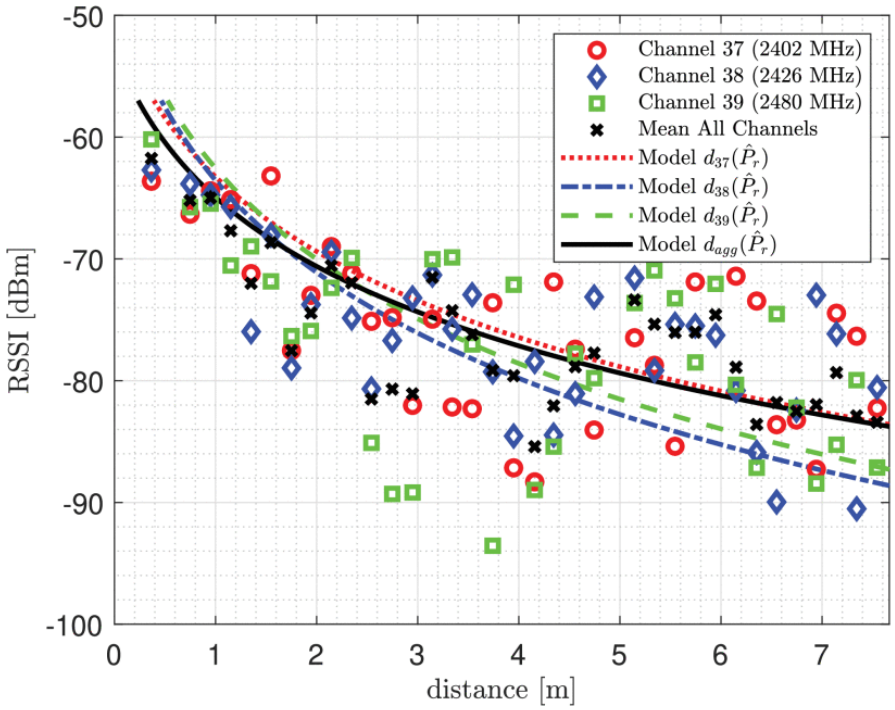


Figure 1.6 Visualization of mean RSSI measurements for different advertising channels over multiple distances. Figure taken from [14].

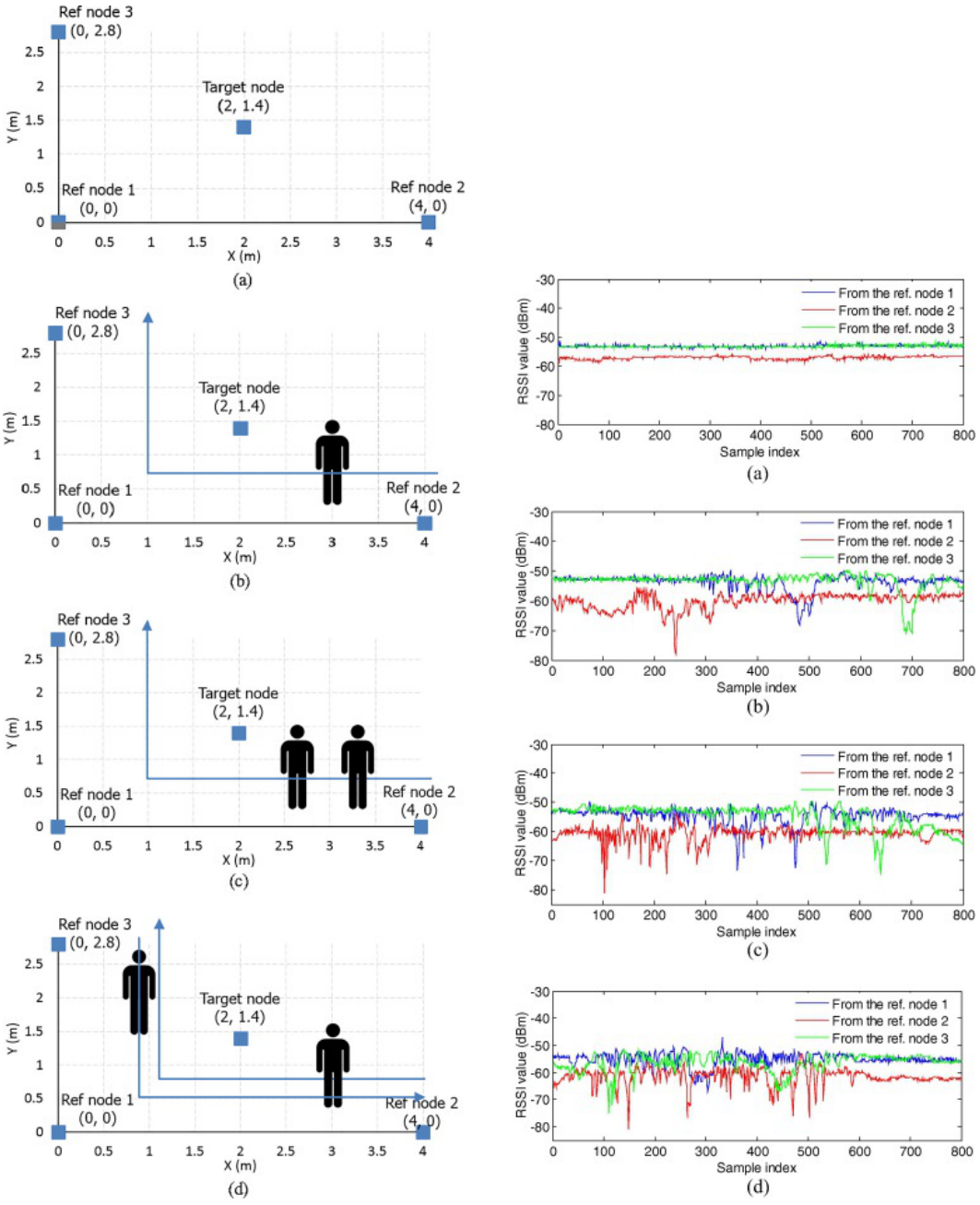
Although this section focused on BLE advertising channels, the same is true for all other channels, used for data transfer.

1.4.5 Human movement

Indoor localization systems often need to work in crowded areas, where human movement is constantly obstructing the radio signal path between a transmitter and a receiver. This section further explores the effects of human movement on measured RSSI.

In [15] researchers created a setup with 3 reference nodes (sources) and one target node. Then they proposed 4 different scenarios (shown in figure 1.7 (a)) and measured RSSI from all 3 sources.

As can be seen in figure 1.7 (a), in test scenario 1 the measured RSSI is almost still. There are barely any observable fluctuations. On the other hand in test scenarios 2, 3, and 4 where a human movement obstructing the sources is present, there are many significant fluctuations of RSSI. As seen in figure 1.7 (b), three main drops of RSSI correlate with a human moving past each of the sources. In scenarios 3, and 4 where two people are passing around sources, there are at least 6 drops of RSSI (2 for each source). This clearly shows that human movement negatively affects the value of RSSI. In [16] the authors estimated the attenuation of RF signal between the human chest and back to be 19.2 dBm. All this evidence points to, humans being a major disturbance for measuring RSSI.



(a) Scenarios of human movement.

(b) Measurements of RSSI.

Figure 1.7 Scenarios for exploring effects of human movement on RSSI and corresponding measurements. Figure taken from [15].

Chapter 2

Time series smoothing

In a natural environment, the signal often gets contaminated with noise, which makes the signal corrupted, is this case filters come in handy! As mentioned in [17] filters have two uses, the first one is called signal separation, and its goal is to separate the desired signal from collected noise. The other one is called signal restoration, which basically means a reconstruction of a signal that has been distorted in some way, for example by faulty measuring equipment, improper calibration, etc. In other words, the overall goal of filters is to help extract useful information from measurements, that might be overshadowed by noise or distortion. This chapter will introduce multiple signal filters suitable for smoothing RSSI measurements.

2.1 Sliding-window filters

In signal processing one of the most common approaches to smoothing is using a sliding window. The window is slid along time series data and takes the average, or weighted average of whichever elements are currently in the window as explained in [18].

2.1.1 Simple moving average

The moving average filter is the most common filter. As mentioned in [19] it is quite popular with investors, who use it as a part of their technical analysis. It also is very intuitive and easy to efficiently implement. The authors of [20] state that if using the recursive algorithm the SMA is the fastest digital filter available. Despite its simplicity, it has been proving itself to be quite an excellent choice in numerous applications, because the SMA provides a very good solution for reducing noise in the signal. The following equations show two implementations of SMA. The x_i stands for the i-th element of input, y_i stands for the i-th element of filtered output, and the M is the width of the moving window. According to [21] the larger the M smoother the filtered output (see figure 2.1). The formula for SMA:

$$y_i = \frac{1}{M} \sum_{j=0}^{M-1} x_{i-j}. \tag{2.1}$$

The equation shows a formula for left-handed SMA. The main advantage of this approach is that it can be calculated in real-time. The disadvantage is that the output is delayed.

$$y_i = \frac{1}{M} \sum_{j=a}^{b} x_{i+j}, \text{ where } a = \frac{-(M-1)}{2}, \; b = \frac{M-1}{2}. \tag{2.2}$$

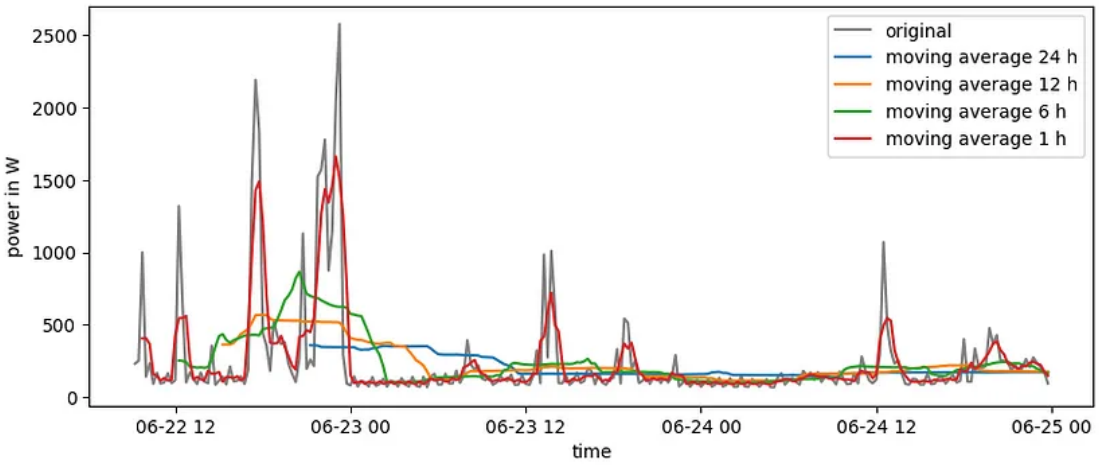


Figure 2.1 Left-hand simple moving average window of different sizes. Figure taken from [22].

The formula (2.2) shows a formula for symmetric SMA. Here the window is centered around the output signal. Symmetric SMA requires M to be odd, so the window is symmetric around the x_i. This solves the delay problem of left-handed SMA. But on the other hand, the filter is not real-time since you have to know the data points that occur after the x_i. So in applications, which need real-time filtering the left-handed SMA is preferred.

2.1.2 Multi-pass moving average

Another approach to smoothing time series is to take advantage of SMA and run it multiple times over collected data. The impacts of such an approach can be seen in figure 2.2 which shows how the window changes over multiple passes. Two passes create a triangular window. After more passes, the window starts to resemble a Gaussian, this is further explained [20]. This approach comes with some benefits, like smaller amplitude near the ends, or smother curves. On the other hand, iterating through data multiple times comes with the cost of extra time.

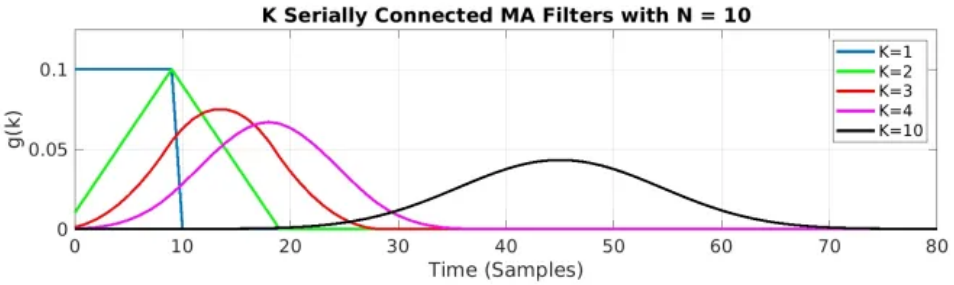


Figure 2.2 Left-hand simple moving average window of length 10 after K passes. Figure taken from [18].

2.1.3 Weighted moving average

Last, the section introduced a multi-pass moving average and showed multiple passes converging to the Gaussian window. In some cases, it is a good idea to have a non-uniform window. Some

requirements for the window are presented in [23]: individual weights in the window are defined by a kernel function and they have to sum up to 1 however, this does not mean that the weights have to be between 0 and 1 and the window itself also has to be symmetrical. Often used kernel is indeed Gaussian, shown in formula (2.3). According to [24] other common kernel functions are often based on *sinc* function. The biggest downside of all kernel filters is their slower execution speed compared to SMA as mentioned in [20].

$$K_{\text{gaussian}}(x, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}. \tag{2.3}$$

2.1.4 Savitzky-Golay smoothing

The Savitzky-Golay filter is essentially once again a moving average filter. The main idea is to locally use least-squares polynomial approximation. It proved itself to be a viable choice for smoothing time series. The original intention of Savitzky and Golay was to smooth noisy data obtained from chemical spectrum analyzers as mentioned in [25]. Nowadays, it has found a use in many applications such as electrocardiogram processing [26], or image processing [27], in its generalized two-dimensional form. In [28] they argue that one of the downsides of the Savitzky-Golay filter is that it is prone to artifacts near boundaries.

The Savitzky-Golay is explained in [25]. The smoothing uses a sliding window of size $2M+1$, so the overall size of the window is odd. Then a polynomial of chosen degree N is fitted, in a way that minimizes mean-squared error with the original data. A higher degree of N tends to reduce mean-squared error while fitted but leads to overfitting of the data, which means worse smoothing. The smoothed output value is obtained at the central point of the fitted polynomial.

2.2 Exponential smoothing

Another approach to time series smoothing is exponential smoothing. *"Exponential smoothing originated in Robert G. Brown's work as an OR analyst for the US Navy during World War II."* [29, p. 638] It proved itself to be a relatively simple but robust and efficient smoothing technique. The [30] introduces three exponential smoothing techniques: simple exponential smoothing, Holt's exponential smoothing, and Holt-Winters exponential smoothing.

2.2.1 Simple exponential smoothing

In [31] the authors explain the benefits of simple exponential smoothing. One of the benefits of is, that it requires only little computation. When the collected data do not show cyclic patterns it can be very efficient. To calculate the smoothed value only the previous smoothed value and current measurement are needed. Despite this, the smoothed value contains information about all previous smoothed values. The formula for simple exponential smoothing:

$$y_i = \alpha x_i + (1 - \alpha) x_{i-1}. \tag{2.4}$$

Here, the y_i represents smoothed output value, and x_i is the actual measured value from the time series. The $\alpha \in [0, 1]$ is the only parameter in this smoothing technique. If $\alpha = 0$, then all smoothed values will be equal to the initial smoothed value y_0. If $\alpha = 1$, then all smoothed values will be equal to the corresponding value in the original time series. As long as $\alpha \neq 1$, the y_1 contains information of all previous smoothed values.

To get the algorithm started the initial value y_i needs to be selected since it is not known. For this, there are several options, the first one is to set the initial smoothed value to the first value in the original time series, making $y_0 = x_1$. The other option is to set it to the mean of

the first n values. In [32] the authors claim that a good number for n might be 5 or 6. But the best practice is to set n empirically for each application.

Another important decision is the choice of the α smoothing parameter. In [31] the authors discuss the effects of α on smoothing. When the value of α approaches 0, the smoothed output heavily relies on the previous smoothed value. On, the other hand if the α is close to 1, the importance of previous smoothed values descends very fast. (Effects of various values of α can be seen in figure 2.3) So the α parameter should be chosen carefully with regard to demanded results, whether is preferred to highlight the overall trend in data or to just gently regularize it. In [33] the authors suggest choosing α so that smoothed output minimizes MSE with the original time series or using Levenberg–Marquardt algorithm. It is also crucial to keep in mind that in time series with a non-equidistant sampling, the α parameter should be modified each step to compensate for the irregular intervals between measurements as mentioned in [34].

Figure 2.3 Simple exponential smoothing with different values of alpha. Figure taken from [35].

2.2.2 Double exponential smoothing

In [36] the authors introduced double exponential smoothing also known as Holt's exponential smoothing. Is very popular in economics since is capable of anticipating trends in data and allows forecasting. The formula:

$$\begin{aligned} y_i &= \alpha x_i + (1-\alpha)(y_{i-1} + t_{i-1}), \\ t_i &= \beta(y_i - y_{i-1}) + (1-\beta)t_{i-1}. \end{aligned} \tag{2.5}$$

In the formula, x_i is the input, y_i represents smoothed output and t_i is the trend. In [37] the authors recommend initializing the t_i to zero. α and β are smoothing parameters. Smoothed output for different values of α and γ is shown in figure 2.4.

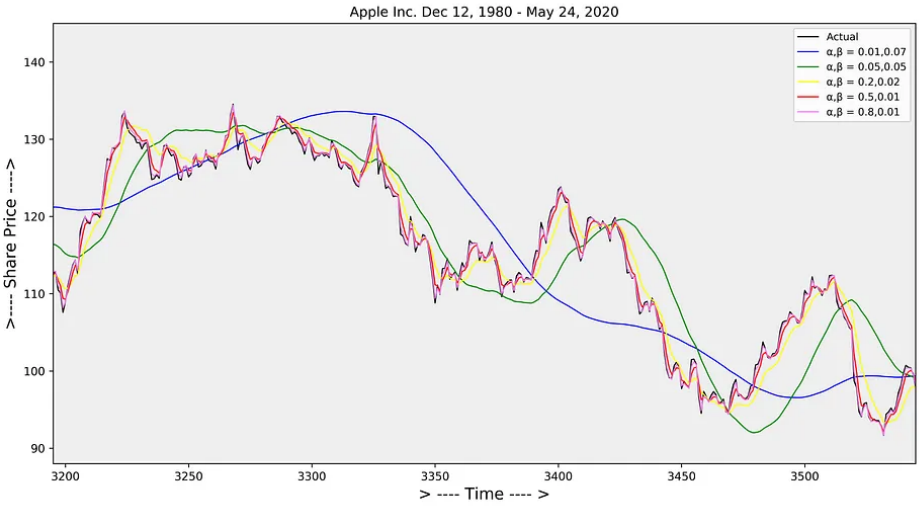


Figure 2.4 Double exponential smoothing with different values of alpha and beta. Figure taken from [35].

2.2.3 Triple exponential smoothing

Also known as Holt-Winters' method. It applies exponential smoothing three times to capture not only trends, like double exponential smoothing, [1] but also the seasonality of data. There are two models of computation, one is in an additive manner and the other one in a multiplicative manner to capture different types of seasonality. The [38] shows the formula for triple exponential smoothing in a multiplicative manner:

$$\begin{aligned} y_i &= \alpha \frac{x_i}{s_{i-L}} + (1-\alpha)(y_{i-1} + t_{i-1}), \\ t_i &= \beta(y_i - y_{i-1}) + (1-\beta)t_{i-1}, \\ s_i &= \gamma \frac{x_i}{y_i} + (1-\gamma)s_{i-L}. \end{aligned} \tag{2.6}$$

And [36] shows the formula in an additive manner:

$$\begin{aligned} y_i &= \alpha(x_i - s_{i-1}) + (1-\alpha)(y_{i-1} + t_{i-1}), \\ t_i &= \beta(y_i + y_{i-1}) + (1-\beta)t_{i-1}, \\ s_i &= \gamma(x_i + y_i) + (1-\gamma)s_{i-L}. \end{aligned} \tag{2.7}$$

In the equations above x_i represents the input, y_i represents smoothed output, t_i is the trend and s_i is seasonality. The $\alpha, \beta, \gamma \in [0, 1]$, are estimated in such a way that the MSE error is minimized. The L represents the number of periods in a season.

To initialize triple exponential smoothing at least one complete season's data is needed to determine s_{i-L} as mentioned in [38]. The [39] introduces way of determining the initial value for trend factor t_0 and season factor s_i:

[1] Although the [33] claims that triple exponential smoothing is more reliable for parabolic trends than double exponential smoothing

$$
\begin{aligned}
t_0 &= \frac{1}{L}\left(\frac{x_{L+1}-x_1}{L}+\frac{x_{L+2}-x_2}{L}+\ldots+\frac{x_{L+L}-x_L}{L}\right),\\
s_i &= \frac{x_i}{A_{L-1}},\ \text{where } 0 \le i \le L-1 \text{ (Multiplicative)},\\
s_i &= x_i - A_{L-1},\ \text{where } 0 \le i \le L-1 \text{ (Additive)},\\
A_{L-1} &= \frac{1}{L}\sum_{j=0}^{L-1} x_j.
\end{aligned}
\tag{2.8}
$$

Figure 2.5 shows the triple exponential smoothing with various combinations of smoothing parameters.

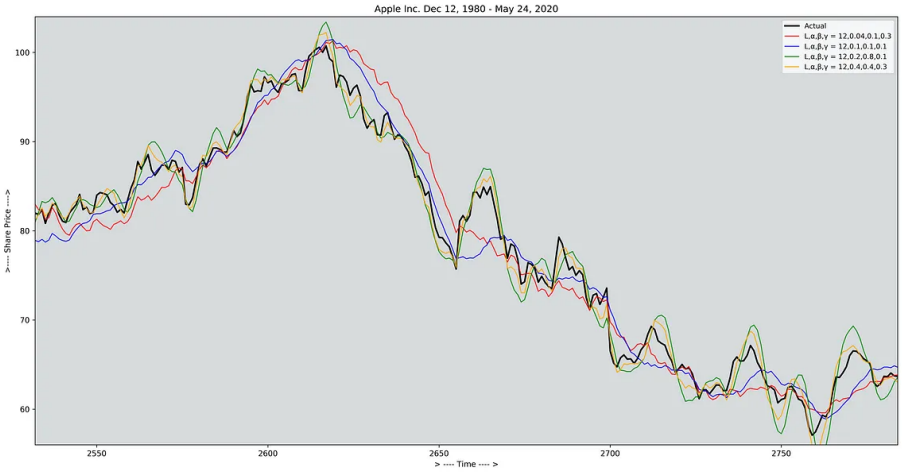


Figure 2.5 Triple exponential smoothing with different values of alpha, beta, and gamma. Figure taken from [35].

2.3 Review of RSSI preprocessing for positioning applications

Smoothing is used in RSSI preprocessing to achieve two main goals: reduce the random noise and highlight trends that correlate with movement. The last few sections introduced several time series smoothing methods suitable for these tasks. But whenever you do research it is a good idea to look not only at general concepts but also at the previous work that has been done in the field. So let's look at how researchers in the past tackled the RSSI preprocessing.

In [40] the authors used BLE RSSI for indoor localization, focused on the hospital environment. They tested the moving average filter, moving median filter, and channel separation preprocessing. The moving average outperformed the moving median smoothing in every scenario. So it was used to smooth data obtained from each channel, in the channel separation step. The channel separation technique was used on advertisement channels (section 1.4.4). The goal was to separate the RSSI from the 3 advertisement channels, into clusters, containing only RSSI from one channel, using the K-means clustering. Then only the data from the given channel were used for positioning estimation. The RSSI from channel 2 gave the best result in minimizing

RMSE and outperformed the model using the moving average on all channels.

In [41], the authors deal with randomness in data by implementing a weighted filter. They take advantage of the fact that if a large number of measurements is taken, their randomness is relatively dependent and the distribution of RSSI at a certain point resembles a Gaussian distribution. This can be seen in figure 2.6. One solution they offered is using a weighted sliding window smoothing. Their formula:

$$y_i = 0.5x_i + 0.2x_{i-1} + 0.2x_{i-2} + 0.1x_{i-3}. \tag{2.9}$$

This formula was found empirically for their use case but, the overall idea can be taken and used for other implementations.

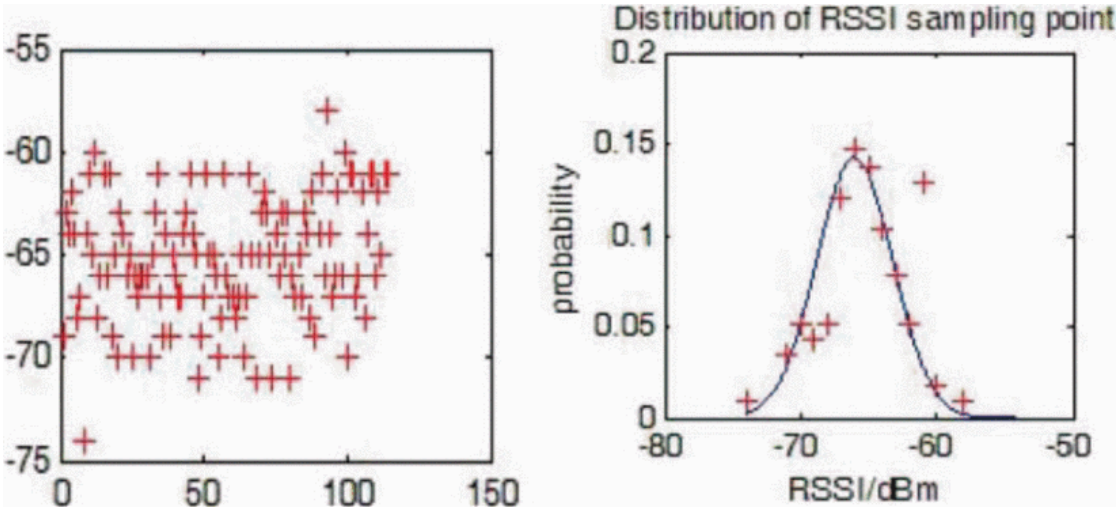


Figure 2.6 Randomness of RSSI at a fixed distance can be thought of as Gaussian distribution. Figure taken from [41].

In [42], the authors also noticed that errors in experimental measurements follow Gaussian distribution, thus they used the Gaussian filter. But they found out that after smoothing the RSSI values still show great variations, which would compromise the distance estimation algorithm. As a solution to this, they proposed using the well-known moving average filter after the Gaussian filter. So the large fluctuations were eliminated by the Gaussian filter, and then the moving average dealt with less apparent noise in the data.

Chapter 3

Data exploration and analysis

This chapter will explore data provided by 2N TELEKOMUNIKACE a.s, to get some insights. Then the chapter will continue with an analysis of the measurements using theoretical knowledge introduced back in previous chapters.

3.1 Introduction

The data sets were obtained from 2N TELEKOMUNIKACE a.s, who collected the data. They made measurements of RSSI, from IP intercoms, using several smartphones. IP intercom is a remote unlocking door unit, equipped with BLE technology, which enables wireless connection with modern mobile phones. The IP intercom can take advantage of this, and trigger a door opening based on the position and movement of an authorized mobile phone. The aim of the measurements was to collect RSSI data from such approaches and leavings so a proximity-based door-unlocking algorithm, can be developed. For this, the researchers at 2N TELEKOMUNIKACE a.s proposed various test scenarios in distinct environments, which accurately resembled real-world circumstances. Then they collected data using numerous models of smartphones, from the most popular mobile phone brands, to achieve certain diversity in the measurements, hoping it may improve the positioning algorithm. One, very effective way of improving the algorithm is to focus on data preprocessing. So the goal is to explore whether and how different devices, models, and scenarios affect RSSI values and how they interconnect with each other. These insights might help to better understand the relationship between measured RSSI and proximity and also may help to find a method of normalizing RSSI, in a way that would improve the door-unlocking algorithm's performance.

3.2 Overview of used technologies

All of the analysis was done using Python 3.11.3. Since it offers a variety of built-in tools and has plenty of data-oriented packages. For instance, Numpy, Pandas, and Scipy were used to store and preprocess the data, perform array manipulations, and perform scientific calculations, necessary for the analysis. It is desirable to use these packages since it not only speeds up the development, but also the calculations, because they are optimized in low-level languages, and therefore performed much quicker, than if done with just Python. Matplotlib and Seaborn were used for the visualizations. Jupyter was used since it provides a clear and simple environment that simplifies the data analysis tasks. Git was used to remotely store the project and better organize the development process. A complete list of all used packages and their respective versions can be found in the attached requirements.txt file.

3.3 Data loading and preprocessing

2N TELEKOMUNIKACE a.s provided two compressed files in .7zip format, with gathered data. The first one contains data collected in controlled scenarios by researchers at the company. The other consists of data collected by users.

The user data consists of countless JSON files. [1] Some of them contain measurements of RSSI from mobile phones and some contain information from door units. The files are named without a clear pattern, which makes it impossible to quickly understand what each file contains. This complicates data processing. Additionally, there is no info about the mobile device or environment in which the data were gathered due to the anonymization of the data. This also fairly limits their usability for analyzing relationships between measured RSSI and external factors, which is among the main goals of this thesis. Therefore, this chapter will focus on further exploring the data collected by researchers in a controlled environment.

The data collected by researchers are categorized by test scenarios:

- **arrival0**, where researchers came up to the door unit,
- **arrival3**, where researchers approach towards the door unit and stop 3 m away,
- **arrival5**, where researchers approach towards the door unit and stop 5 m away,
- **arrival7**, where researchers approach towards the door unit and stop 7 m away
- **leaving3**, where researchers started moving away from the door unit and stopped 3 m away,
- **leaving5**, where researchers started moving away from the door unit and stopped 5 m away,
- **run**, where researchers ran up to the door unit.

Each scenario corresponds to one folder in the dataset, so there is a total of eight folders. Let's look at data from arrival0 as an example. As mentioned in the paragraph about user data, there are two types of files, based on what they contain. This is the same for arrival0. The first type contains time series of RSSI measurements from smartphones (this type of file will be further referred to as a measurement file). The other one contains information from door units (this type of file will be further referred to as an access_log file). Other folders contain analogical data for different scenarios. But there are a few differences in the format of a measurement file. It depends on the kind of operating system of the mobile phone which was used to take the measurements. The smartphones running on Android operating systems have their measurement files in CSV format, but on the other hand, iPhones, which use the iOS operating system have their measurements in .txt files. These .txt files contain data in JSON format. The access_log files are in CSV format for all operating systems, but there in key differences in their contents.

3.3.1 Contents of data files

The following lists show the content of obtained data files. The bold text on the left-hand side stands for the name of the feature. If there is an arrow, it means that the name was changed to the one arrow is pointing to. The text in parentheses on the right-hand side is the type of the feature. If there is an arrow, it means that the original type was changed to the one arrow is pointing to. Note that the access_log files contain more features than listed, but those are considered irrelevant, so they are not included.

[1] Some of them are over 300 MB in size!

Android measurement

- **date**: time when current measurement was taken (object → datetime)
- **rssi**: value of measured RSSI (int64 → int8)
- **source**: packet type (object → bool)
- **startOpeningTime** time when door started opening (object → datetime)

iOS measurement

- **identifier** identifier of approach (object → category)
- **rssi**: value of measured RSSI (int64 → int8)
- **rssiSource** → **source**: packet type (object → bool) (This column is optional)
- **threshold**: threshold to trigger door opening (float64 → float32)
- **timestamp** → **date**: time of measurement (object → datetime)
- **acceleration**: information about acceleration
  1. **x**: coordinate (float64 → float32)
  2. **y**: coordinate (float64 → float32)
  3. **z**: coordinate (float64 → float32)
- **rotation**: information about rotation
  1. **x**: coordinate (float64 → float32)
  2. **y**: coordinate (float64 → float32)
  3. **z**: coordinate (float64 → float32)

Android access_log

- **connectionTime**: time when the first connection packet arrived (object → datetime)
- **correctResult**: information whether the door opened correctly (bool)
- **device name**: name of the intercom taking the gathering the data (object → category)
- **endOpeningTime**: time when the door unit stopped opening (object → datetime)
- **rssi threshold**: RSSI value that must be exceeded to trigger opening (int64 → category)
- **scannedTime**: time when first the Advertisement packet arrived (object → datetime)
- **startOpeningTime**: time when the door unit started opening (object → datetime)

iOS access_log

- **connectionTime**: time when the first connection packet arrived (object → datetime)
- **deviceName**: name of the intercom taking the gathering the data
- **endOpeningTime**: time when the door unit stopped opening (object → datetime)
- **isCorrectResult**: information whether the door opened correctly (bool)
- **scannedTime**: time when first the Advertisement packet arrived (object → datetime)
- **startOpeningTime**: time when the door unit started opening (object → datetime)
- **threshold**: RSSI value that must be exceeded to trigger opening

3.3.2 Transformation to HDF5

To speed up future IO operations and standardize access it is desirable to reorganize and transform obtained data into a more efficient format, some that will fulfill these needs. HDF5 is an excellent choice for this task since it is able to handle large amounts of data efficiently and also enables keeping the original structure. It provides two basic objects *groups* and *datasets*. Groups provide internal structure and datasets contain data, this is further explained in [43]. So the data were transformed into a such file. The overall structure remained the same as before. There are eight groups each one corresponding to one scenario. Each group contains measurement files and access_log files. The iOS measurement files are split into three distinct files, this division makes the data better organized. The first one contains RSSI values over time. The second one contains measurements of acceleration. And the last one contains measurements of rotation.

Data from CSV and JSON documents were loaded into Pandas DataFrame objects, where each measured subject (*startOpeningTime*, *source*, etc.) corresponds to one column. Then all values were transformed, suitable types were selected and everything was downcasted so the DataFrames take less space. Some columns got renamed to get unified names in Android and iOS measurements. Everything can be seen in the section 3.3.1. DataFrames with measurements from mobile phones got sorted by *date*, and the *date* column was set as an index. After this preprocessing step, DataFrames were saved into one HDF5, each to their respective group.

3.4 Measurements analysis

The last section introduced researched dataset and its structure. To explore the dataset let's explore some measurements by creating a plot.

Figure 3.1 shows the last 500 measurements, taken with Zenfone8 in the arrival0 scenario. That means that researchers took the Zenfone8 and approached right next to the door unit. There are 3 observable peaks in the RSSI value, and it seems they are coming in approximately the same intervals. It looks like there is a seasonal pattern. It became clear that the researchers were making multiple approaches, and saving them to one file in one continuous time series, this caused the seasonal pattern to occur. Each peak corresponds to the researcher being close to the door unit.

Each measurement taken can be easily distinguished using the *startOpeningTime* column that enables the identification. This is shown in figure 3.2.

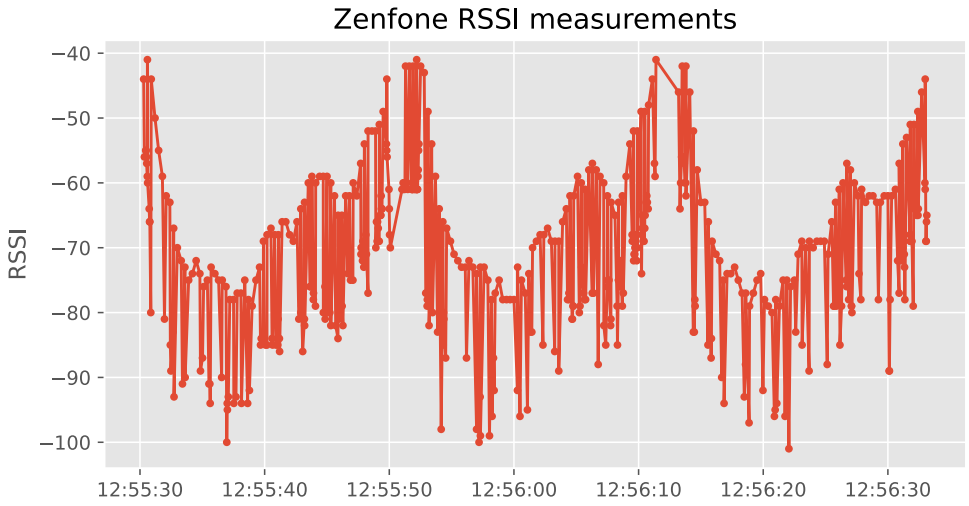


Figure 3.1 Arrival0/Zenfone8 - Last 500 measurements.

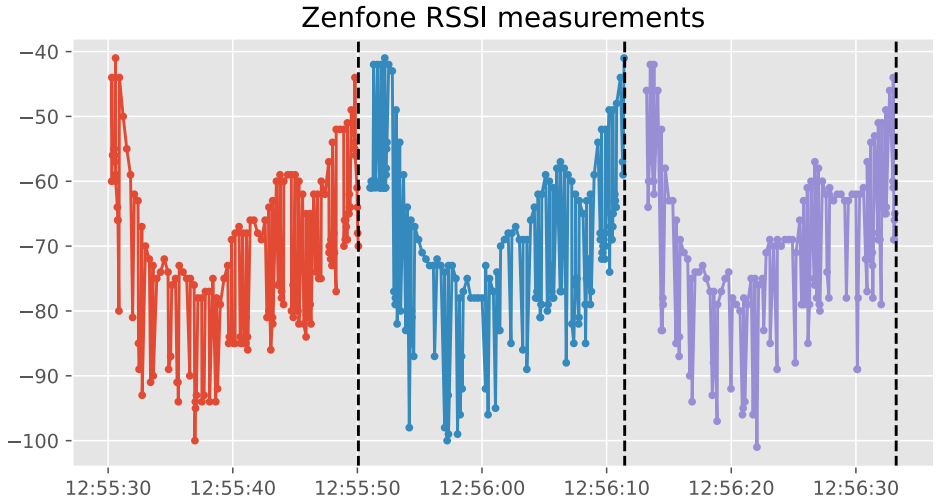


Figure 3.2 Arrival0/Zenfone8 - last 3 approaches with opening times.

Dotted lines represent *startOpeningTime*. Each one of the three measurements has its distinct color. The measurement should start 20 seconds before *startOpeningTime*. In this case, each measurement starts right after the previous peak, then when the researcher returns to his initial position, it quickly drops. Then it remains about the same for a short period. And as he starts approaching the door unit the RSSI rises until the peak, when the researcher reaches the door. There are 3 observable stages, depending on RSSI value:

- **retreat**: RSSI decreases,
- **the turning point**: RSSI is about the same,
- **approach**: RSSI increases.

For arrival and run scenarios, the most important part of the measurement is the approach. For leaving it is the retreat phase.

Unfortunately, not all approaches and retreats are the same. RSSI is very sensitive to environmental factors as explained in 1.4. This makes it impossible to expect that approaches will look the same in different environments. But since the measurements were taken controlled environment[2], similar conditions for each approach can be expected. This enables investigation of how approaches differ from each other, by test scenario. And also how the different models of smartphones affect the measured RSSI, in the same scenario.

3.4.1 Differences by scenarios

Figure 3.3 shows one approach from each arrival test scenario. In the plot, there are only connection packets since plotting all arrival scenarios with both packets would overshadow the differences in scenarios with noise (see section 3.5 for more information on packets). Note that in all following plots zero represents the *startOpeningTime*.

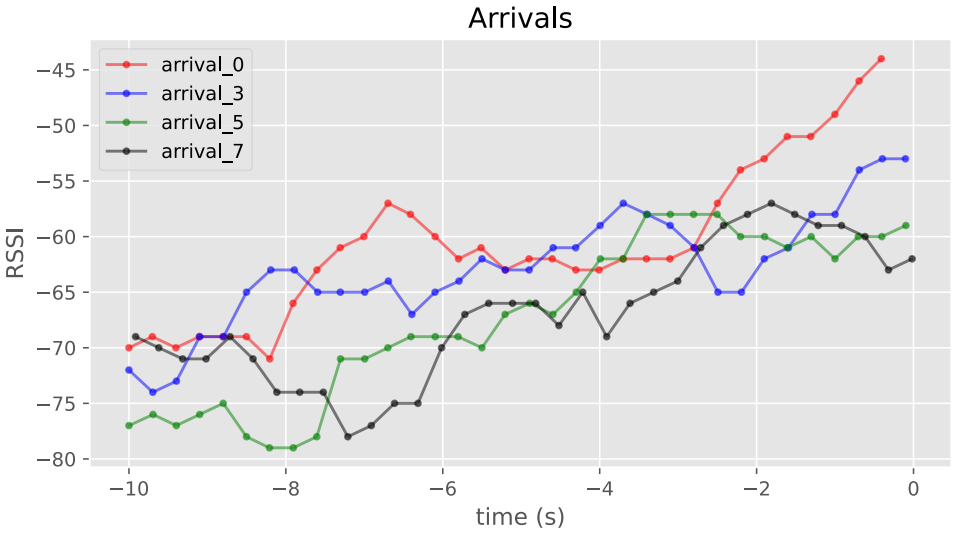


Figure 3.3 Zenfone8 - one opening for each arrival scenario.

As expected, the arrival0 scenario ends up with the highest RSSI. And as the final distance increases the maximum RSSI gets gradually lower. But notice that at the first 6-7 seconds the measurements look about the same and it would be hard to guess which is which. So it is really the last few seconds that make the difference.

Figure 3.4 shows an example of measurements from leaving scenarios. Again further distance from the door unit weakened the RSSI. So leaving5 ends up with a lower RSSI than leaving3.

Figure 3.5 shows dat from the run scenario. Run is a bit specific scenario. The measurement starts about 16 seconds before *startOpeningTime*. Also, there is a significant gap between the first advertisement packet and the first connection packet. The measurements in the first 8 seconds are very sparse. This is caused by the fact that the researchers started far away and were moving wildly across the building towards the door-opening unit while taking these measurements. Moving far away from the door unit, so only a few advertisement packets could reach them. And they came in irregular intervals. 6 seconds before *startOpeningTime*, the connection

[2]Altough, there certainly were some disturbances, causing RSSI to fluctuate, every measurement was taken in the same controlled scenario, so the factors like a reflection of the walls can be neglected, since it is the same for all measurements.

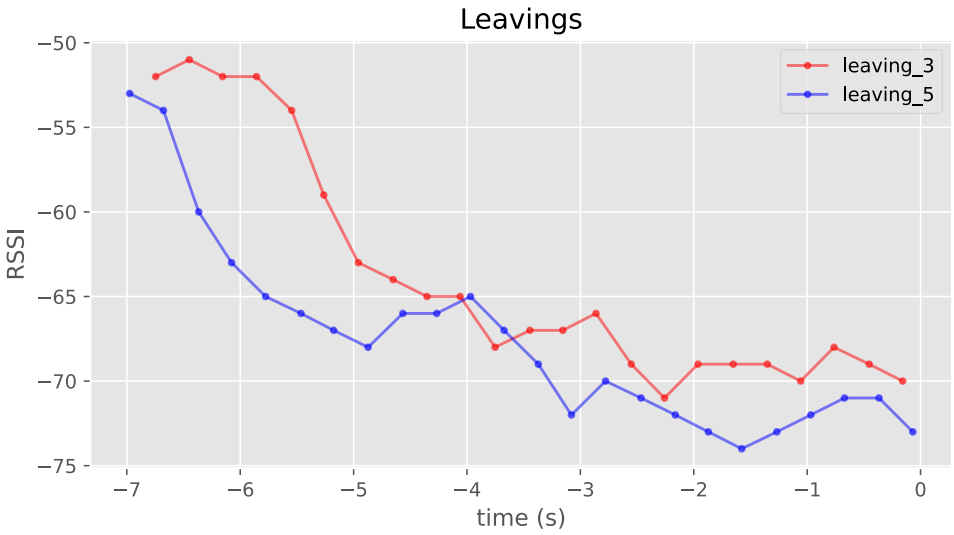


Figure 3.4 Zenfone8 - one opening for each leaving scenario.

packets start coming in and the rest of the measurement looks a bit like an approach from an arrival scenario. But the final RSSI value is much lower than in arrival0, where the final RSSI value was around -45 dBm, but here it is only slightly over -60 dBm.

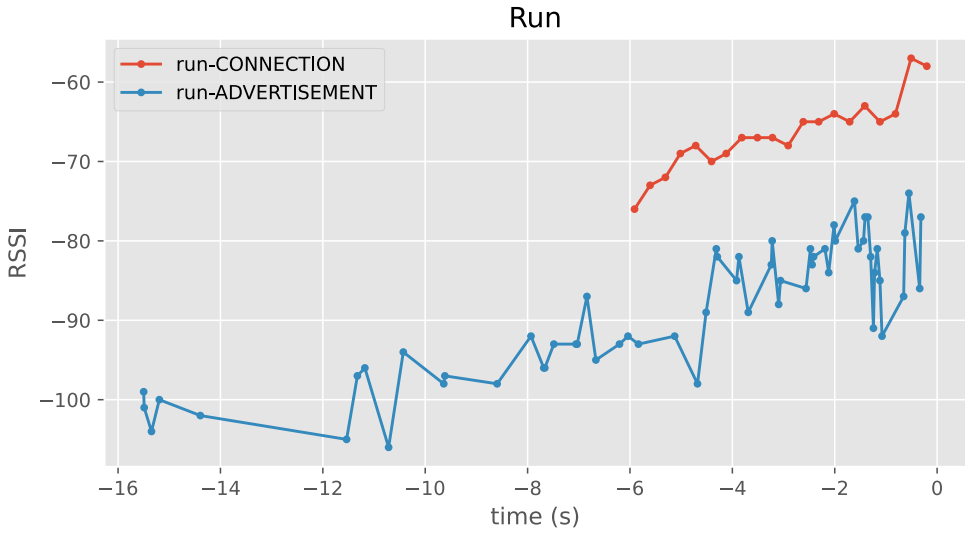


Figure 3.5 Zenfone8 - one opening from the run scenario. Figure is showing both packet types.

3.4.2 Differences by phone models

Another perspective is to look at the single scenario and explore if there are differences between phone models/brands. Nowadays, there are dozens of phone manufacturers, producing countless amount of unique models. They use different hardware with a specific calibration. Materials used for making phones' bodies also differ, they might be consisted of plastic, glass, metal,

leather, etc. The structure of the body and the placement of the BLE antenna also varies from model to model. So let's explore if these differences affect measured RSSI values in any way, and investigate if there is any pattern that would help to find compensation methods, that could be used to normalize RSSI across different models/brands.

Let's analyze approaches from arrival0 scenario. To do this last 7 seconds before opening was analyzed, by calculating the mean value of RSSI in intervals of 500 ms. This is shown in the figure 3.6 for Xiaomi Mi 8 Lite.

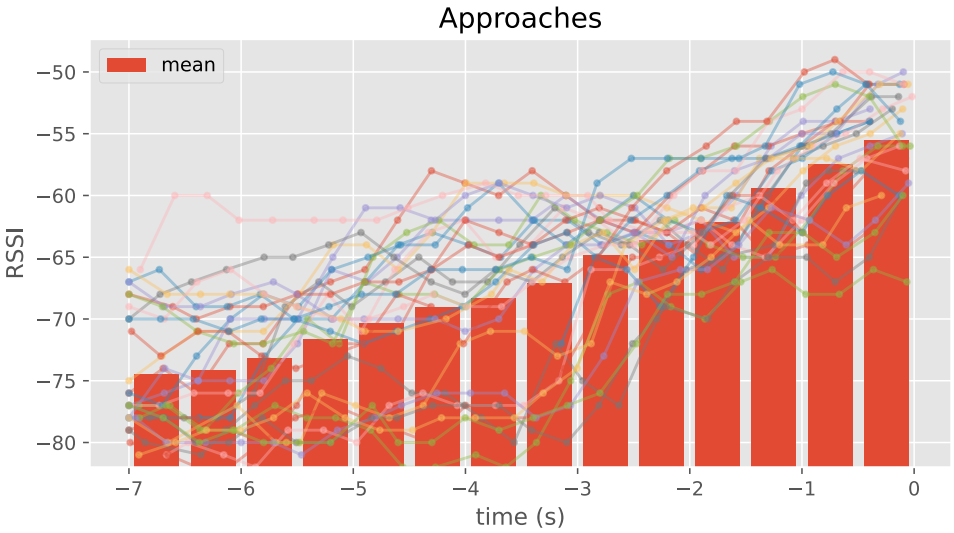


Figure 3.6 Xiaomi Mi 8 Lite - 30 approaches from arrival0 scenario.

The curves nicely resemble approaches. Although values of RSSI at the same time differ from approach to approach the overall trend is clearly linear. As the device is closing on the opening unit the RSSI gets higher.

Let's explore if the RSSI of other models progresses in the same way over time. Figure 3.7 shows the progress of RSSI of phones from 3 different brands.

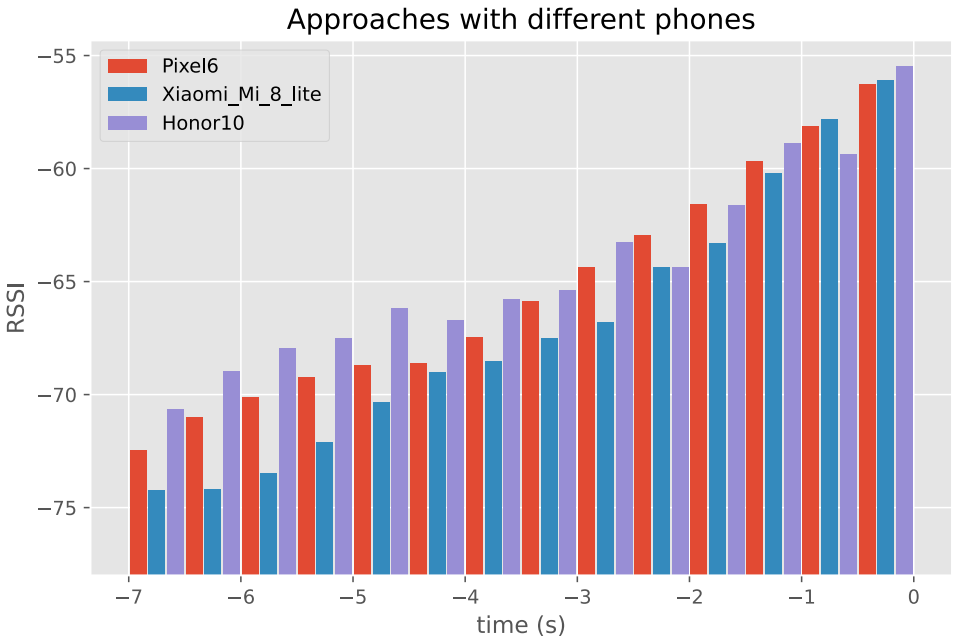


Figure 3.7 Comparison of RSSI in arrival0 scenario, by different phones.

The differences are visible at first sight. All the 3 phones finish with a similar RSSI of about -56 dBM. But the way each one of them gets there is unique. For example, the Honor 10 model starts much higher than the other two models and grows rapidly, but about 4.5 s before opening slows down and is caught by Xiaomi and exceeded by Pixel, only to match them both seconds later. The overall trend for all of the models seems to be linear but for some models like Honor 10, it is quite skewed in some parts of the approach. This happens because differences in hardware and body materials kick in, causing unique attenuation of RF signal for each model. The differences are unpredictable and could only hardly be compensated.

Let's explore if at least the different instances of the same model behave similarly. Figure 3.8 shows approaches with different instances of iPhone 13.

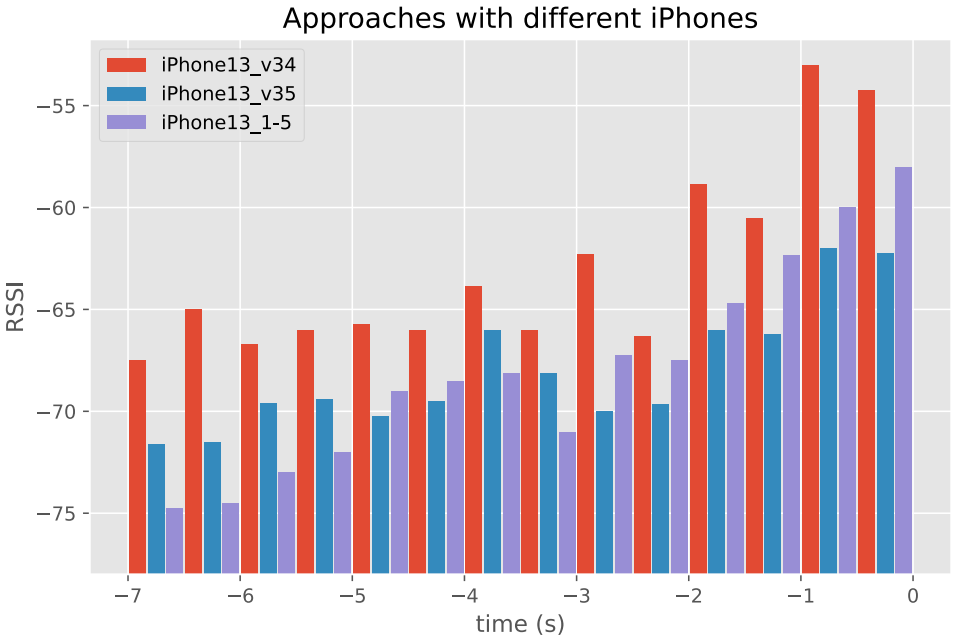


Figure 3.8 Approches in arrival0 scenario using different iPhones13.

There are 3 progressions of RSSI values, from 3 instances of iPhone 13. The means were obtained from 10 approaches since only 10 were done for each phone. This causes the means to fluctuate more than in the previous example. But still, the linear relationship with RSSI is apparent, for all 3 phones.

At first glance, there are huge differences in RSSI in models all over the timeline. The v34 model has a much higher RSSI than two other phones almost for the entire approach. This seems to be unreasonable since they are all the same model.

There are a few possible explanations for this. Either each model has unique hardware calibration which makes it behave differently than other phones of the same model. Or there might have been an unknown variation in the way of taking the measurements.

In any case, it seems that there is no systematic difference in RSSI measurements with different phones, and the variations have to be taken into consideration as random noise.

3.5 Packet types

One of the main factors causing noise in measured RSSI is that different packet types are transmitted with different power levels. The connection packets have generally higher measured RSSI than advertisement packets at the same distance (see figure 3.9). One solution to this problem might be to use only one type of packet for the approach detection algorithm. At some moments during the measurement, only one packet type is available, usually, it is an advertisement. So it seems like a viable option to use only advertisement packets. It might work just fine. But on the other hand, advertisement packets tend to be much have much more noise in them, than connection packets. Therefore the connection packets seem to be more reliable for distance estimation than advertisement packets. So both packets have their upsides and downsides, but if

they are used both together, then once the connection packets start arriving, the data becomes extra noisy. It would be great to find out a way of normalizing these two packet types so they would be in the same power levels.

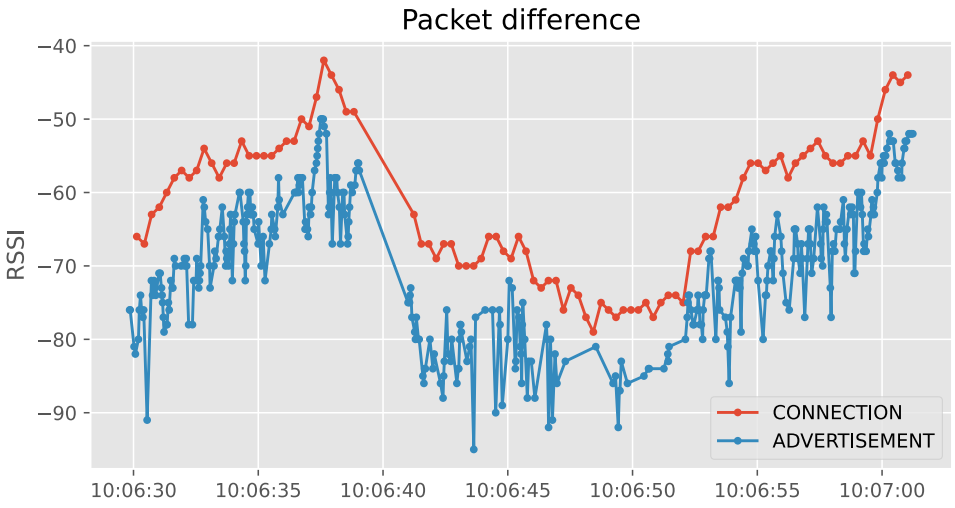


Figure 3.9 Sony Xperia 5 - Advertisement and connection packets have different power levels.

To further explore the possibility of normalizing packet types, it is necessary to know if they carry similar information. Otherwise using both packet types would compromise the model. To investigate this let's calculate the Pearson correlation coefficient between the packet types. The result is shown in the table 3.1.

Table 3.1 Correlation coefficients between advertisement and connection packet for selected phones.

Packet correlations			
Phones	mean	median	var
asuszenfone8 1	0.542	0.802	0.174
oppo_reno3_02	0.790	0.809	0.007
pixel6	0.565	0.677	0.076
samsung_x	0.841	0.872	0.016
xiaomi_mi_8_lite	0.849	0.881	0.011
iphone12_1	0.658	0.750	0.081
iphone13_1	0.691	0.723	0.029
iphonese2020	0.591	0.636	0.069

In correlation estimation for individual phones, all scenarios were considered. And correlation was calculated using overlapping intervals, where both types of packets are present. In these intervals, new RSSI values for both packet types were interpolated at new equidistant time points.

As table 3.1 shows, the median correlation is usually over 0.7, which, according to [44], indicates a strong positive linear relationship. And for all phones, it is over 0.5, which indicates a moderate relationship. The lower correlations might be caused by a small amount of collected data, which might have led to imprecise calculations. But overall, the correlation is high and positive, so normalization can proceed.

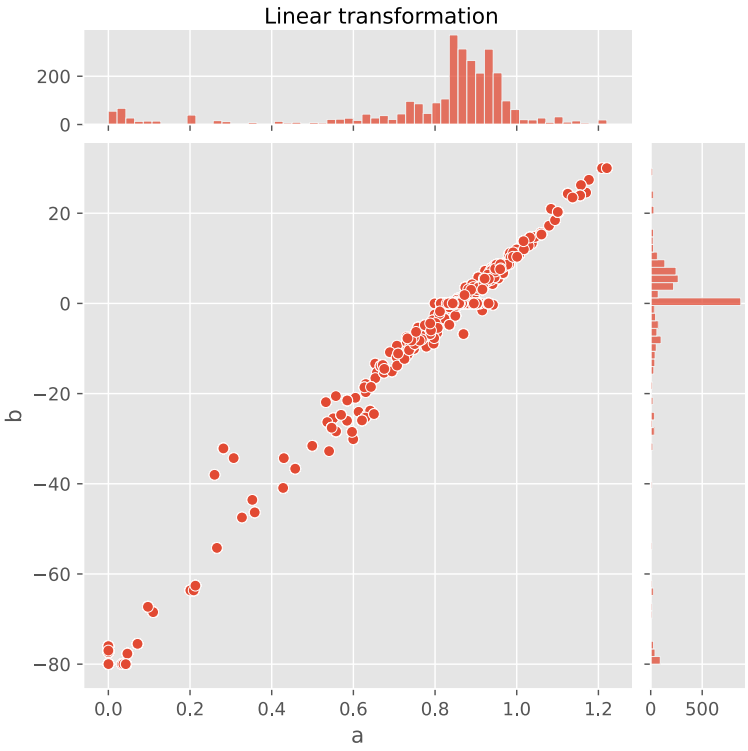
3.5.1 Linear transformation

Since the high correlation indicates a linear relationship between packet types, one of the most intuitive ways of normalization is to find the linear transformation, from one packet to another. So at a time point i the connection packet RSSI y_i can be calculated from advertisement RSSI x_i, as shown in the equation:

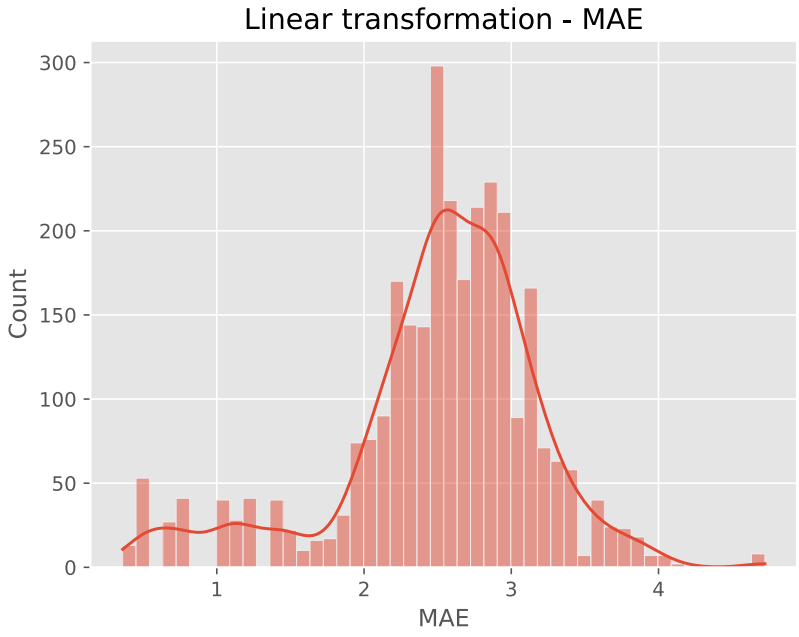
$$y_i = ax_i + b. \tag{3.1}$$

Let's find such a and b, that the MAE error between the time series of the two packet types is minimized. The same trick used to calculate correlation is applied to compute MAE. MAE is obtained from overlapping intervals, where RSSI can be interpolated at equidistant points, thus enabling to get MAE. This can be done for each approach in every scenario. The result is shown in figure 3.10 (a).

The distribution of parameters a and b seems to be pretty crumbled. This makes it hard to select parameters that would perform well in all scenarios. The next section will propose a possible solution.



(a) Values of parameters for linear transformation.



(b) MAE of a linear transformation.

Figure 3.10 Sony Xperia 5 - Scatter plot of the best parameter values for individual approaches and MAE values after the transformation.

3.5.2 Shift transformation

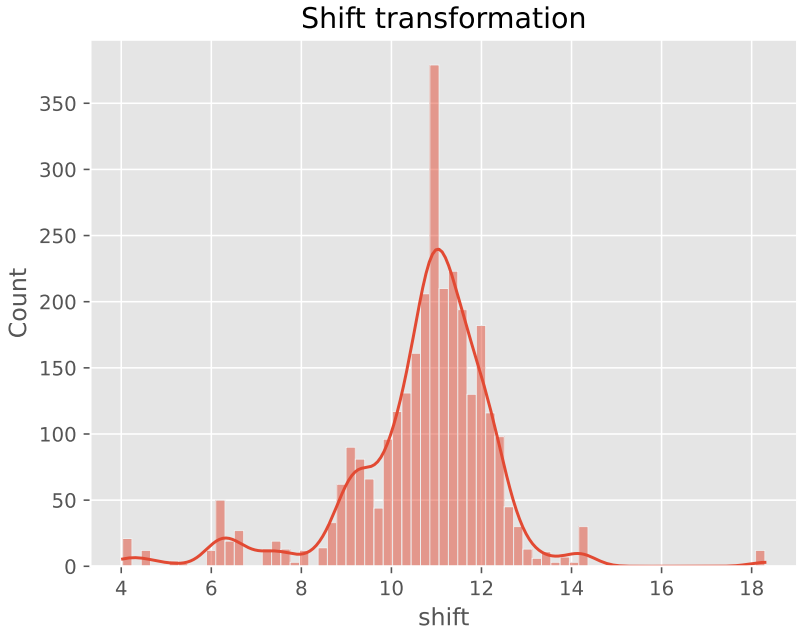
The idea of this transformation is to take the linear one and restrain it, so the $a = 1$ and only the b can be modified[3]. The formula is shown in the equation:

$$y_i = x_i + b. \tag{3.2}$$

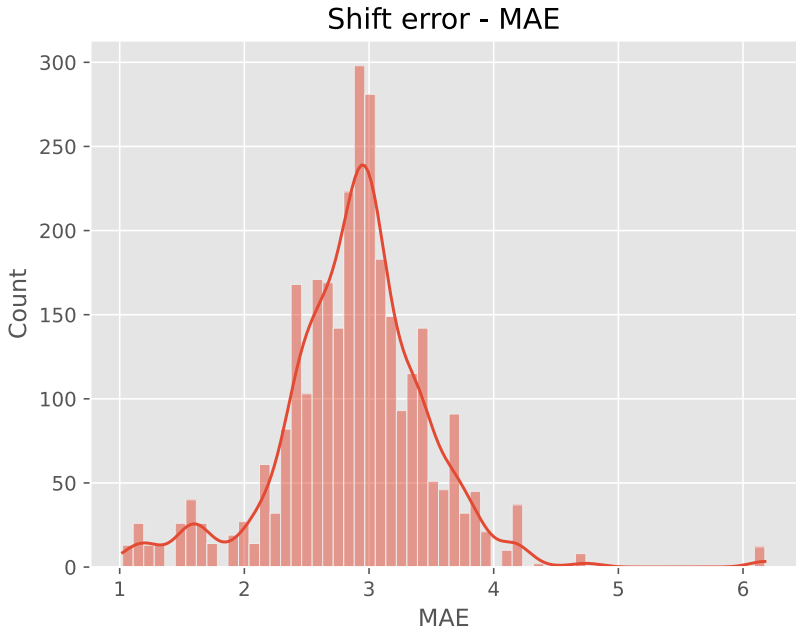
All the transformation can do now, is to shift the advertisement packets' time series up, closer to the connection packets' time series so the MAE is minimized. The same trick as before is used to calculate MAE. This constraint on the linear transformation might help the transformation to generalize better.

As figure 3.11 (a) shows, the shift of the advertisements packets seems, to be normally distributed. Most of the optimal shifts are between 10 and 12. The fact that only 1 parameter is used makes it easier to pick one that will be working for most scenarios. This of course comes at a cost. The linear transformation has more degrees of freedom. Which helps it to fit the target time series better, than just a plain shift. This can be seen by comparing MAE in figures 3.10 (b) and 3.11 (b). It clearly shows that the shift transformation has a higher average MAE than the linear transformation. Since the preprocessing should work for any scenario it is preferable to pick the simpler shift transformation and sacrifice some extra error.

[3]The b in this transformation is often referred to as shift.



(a) Values of shift for transformation.



(b) MAE of a shift transformation.

Figure 3.11 Sony Xperia 5 - Histograms of the best shift parameters values for individual approaches and MAE values after the transformation.

3.6 RSSI smoothing

Introduced in chapter 2, smoothing effectively removes outliers from measured data and highlights trends. Let's implement some smoothing filters and look at how they work for RSSI.

3.6.1 Simple moving average smoothing

One of the most promising and at the same time simplest smoothing techniques. The moving average smoothing is introduced in section 2.1.1, it is very fast to compute and great at reducing random noise in measurements. Figure 3.12 shows the left-handed moving average on RSSI data measured with SamsungS9. The plot in the upper left-hand corner shows the original measured RSSI. It is indeed quite noisy. Especially RSSI from advertisement packets. The other plots show smoothed data with differently-sized moving average windows. It nicely shows how the trends are becoming more visible with the growing window size.

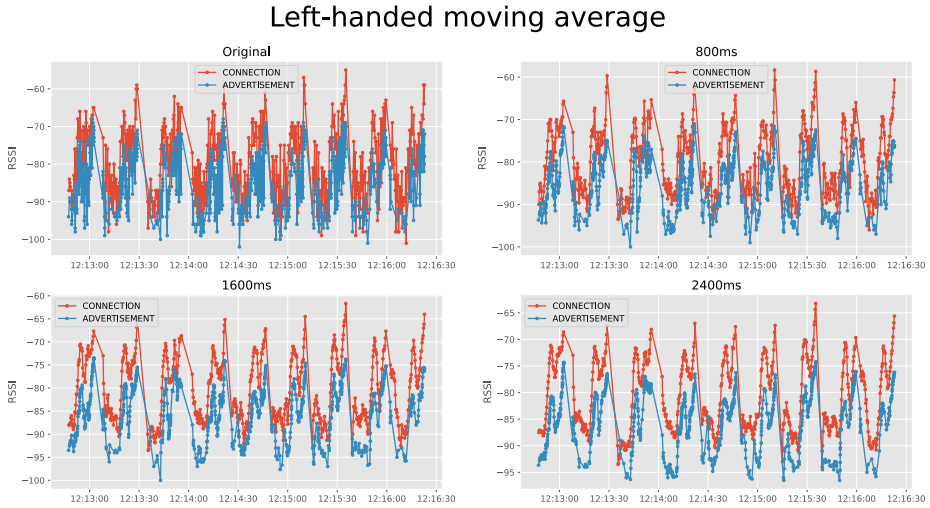


Figure 3.12 SamsungS9 - Left-handed moving average smoothing with differently-sized windows.

3.6.2 Savitzky-Golay smoothing

Another approach to smoothing comes from Savitzky and Golay. Introduced in section 2.1.4, the Savitzky-Golay smoothing can be an efficient way of smoothing. It does a better job in preserving peak with only a few points that moving average. Figure 3.13 shows the Savitzky-Golay on RSSI data measured with SamsungS9.

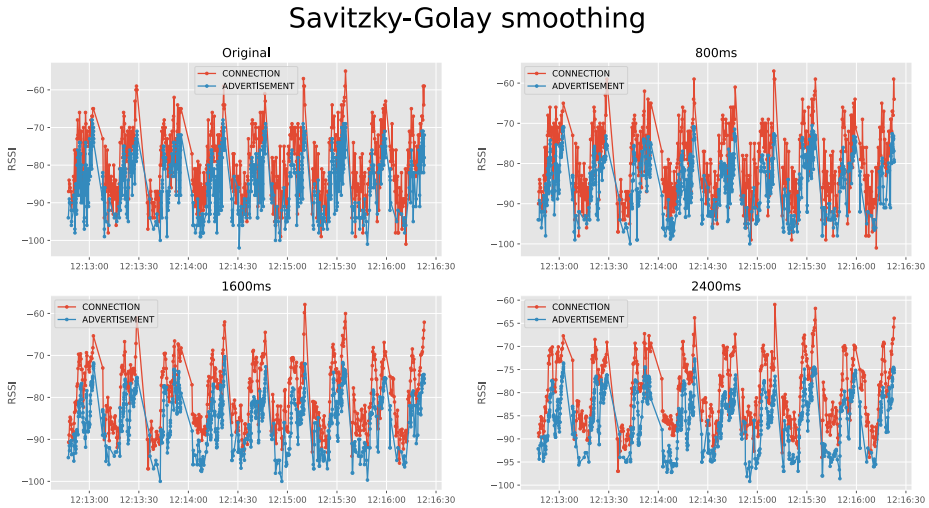


Figure 3.13 SamsungS9 - Savitzky-Golay smoothing with differently-sized windows, with polynomial order of 3.

3.6.3 Simple exponential smoothing

Very fast smoothing technique. The simple exponential smoothing was introduced in section 2.2. One of its main benefits is that it gives more significance to recent measurements. It is also very easy to implement and fast to compute since it needs only the last smoothed value and current measurement to calculate the next smoothed value. The basic version of this smoothing technique works great with evenly spaced values, but it is not suitable for smoothing data with values in irregular intervals. This is one of the downsides. To compensate for this the smoothing factor α has to be adjusted each iteration, to address the different spacing. The proposed formula for calculating the smoothing factor α:

$$\begin{aligned} \alpha_i &= 1 - (1 - \alpha_0)e^{-\frac{t}{\Delta t}}, \\ y_i &= \alpha_i x_i + (1 - \alpha_i)x_{i-1}. \end{aligned} \tag{3.3}$$

The $\alpha_0 \in [0, 1]$ is the initial smoothing factor. Let's assume the smoothing for n consecutive measurements. The $t = t_i - t_{i-1}$ is equal to the difference between the incoming time of the current RSSI measurement t_i and the time of the last RSSI measurement t_{i-1}, where $i \in \{1, ..., n\}$. This means that $(\forall i)((t_i \geq 0 \wedge t_i > t_{i-1}) \implies t \in (0, +\infty))$. The $\Delta t \in (0, +\infty)$ is a parameter that acts like a time window. And the α_i is the final regularization parameter for calculating the i-th smoothed value.

The desired behavior is that the $\alpha_i \in [\alpha_0, 1]$, and is increasing with respect to t and strictly increasing if $\alpha_0 \neq 1$. The proposed formula satisfies these demands. If the t is tiny, which happens if a packet comes just a moment after the previous one, then the α_i should be close to the original α_0. On the other hand, if a packet comes ages after the last one, then the α_i should be close to 1 since the preceding measurements become obsolete. So with the increasing time interval between consecutive packets, the α_i should also be increasing from α_0 to 1. How quickly this happens depends on the time window Δt. Let's check if the proposed formula fulfills these demands:

$$\begin{aligned}\lim_{t\to 0}\left(1-(1-\alpha_0)e^{-\frac{t}{\Delta t}}\right) &= 1-(1-\alpha_0)e^{0} = 1-(1-\alpha_0) = \alpha_0,\\ \lim_{t\to +\infty}\left(1-(1-\alpha_0)e^{-\frac{t}{\Delta t}}\right) &= \lim_{u\to -\infty}(1-(1-\alpha_0)e^{u}) = 1-(1-\alpha_0)0 = 1.\end{aligned} \tag{3.4}$$

The equations above show that when t approaches zero (packet arrival times close to each other), then the formula for α_i returns α_0. And when t approaches infinity (packet arrival times far away from each other), then the formula for α_i returns 1. So the formula behaves as expected.

Let's investigate how the Δt affects the α_i. Without loss of generality, let the t be fixed to a value, then:

$$\begin{aligned}\lim_{\Delta t\to 0}\left(1-(1-\alpha_0)e^{-\frac{t}{\Delta t}}\right) &= \lim_{u\to -\infty}(1-(1-\alpha_0)e^{u}) = 1-(1-\alpha_0)0 = 1,\\ \lim_{\Delta t\to +\infty}\left(1-(1-\alpha_0)e^{-\frac{t}{\Delta t}}\right) &= 1-(1-\alpha_0)e^{0} = 1-(1-\alpha_0) = \alpha_0.\end{aligned} \tag{3.5}$$

So for fixed t, if the chosen Δt is tiny (approaches zero), then the α_i is close to one for all values t. If the chosen Δt is very large (approaches infinity), then the α_i is close to α_0 for all values t. So the overall rule is, the larger the Δt, the slower the α_i gets to 1 with increasing t. This makes Δt work as a kind of time window for exponential smoothing, and its choice affects the smoothing result.

Figure 3.14 shows this approach to simple exponential smoothing on RSSI data measured with SamsungS9, with different values of Δt.

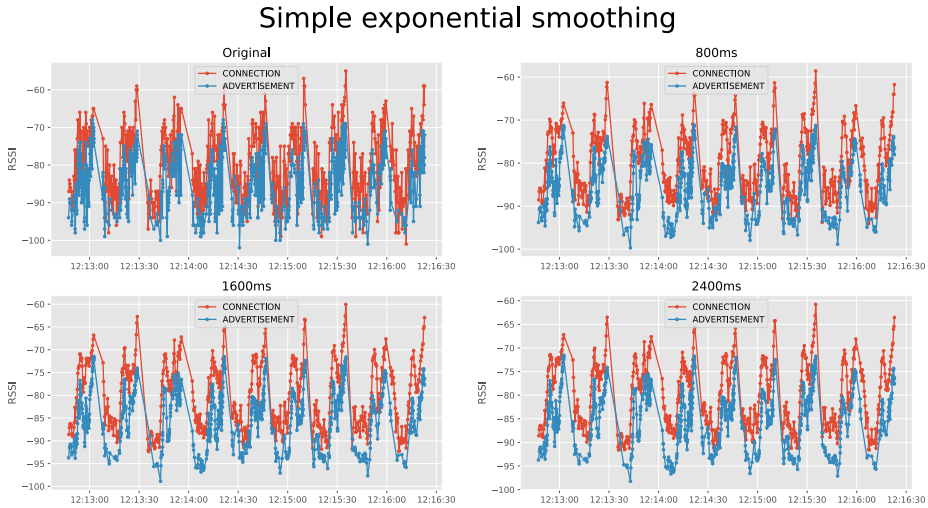


Figure 3.14 SamsungS9 - Simple exponential smoothing with differently-sized windows, $\alpha_0 = 0.2$.

Chapter 4
Experiments

In the previous chapter, the data was analyzed and several well-established smoothing methods were implemented. A way of normalization was also proposed. The goal of this chapter is to evaluate the effects of the smoothing and normalization methods on the remote door unlocking algorithm predictions. And then discuss the results.

The algorithm for door opening was obtained in three configurations: default, secure, and convenient. Each configuration has to fulfill different security standards. For the experiments, the default configuration will be used as an example.

The performance of such a model is measured with accuracy in %, for every scenario individually. The desired behavior is for the door to open in arrival0 and run scenarios. For the other ones, it should stay locked. Table 4.1 shows the original performances of the model for Androids and iPhones:

Table 4.1 Evaluation of default remote unlocking model by individual scenarios. Values correspond to accuracy in %.

Default model performace								
category	arrival0	arrival3	arrival5	arrival7	leaving3	leaving5	leaving7	run
Androids	98.57	18.3	27.34	21.48	92.3	98.33	98.93	80.02
iPhones	91.21	17.6	32.45	45.79	81.21	92.37	92.25	81.01

Values in the table can be used as a reference to evaluate the effects of each preprocessing. Note that the algorithm itself is not modified in any way and only the preprocessing step is changing. The following sections will investigate the effects of several preprocessing techniques.

4.1 Packet normalization

As shown in section 3.5 RSSI data comes in two different packet types, which are highly correlated but have different power levels. Then a normalization technique was proposed in section 3.5.2. Let's investigate how packet normalization affects the algorithm. The expectations are that the algorithm will be more likely to open since the RSSI will generally increase. But it might help to make accurate predictions by minimizing noise created by different packet types. Table 4.2 shows the resulting performance.

The table clearly shows that above mentioned assumption was correct. The accuracy in arrival0 and run scenarios increased. But on the other hand, accuracy in other scenarios decreased. The higher RSSI is causing the algorithm to open more often even if it should not. So the normalization is basically trading-off accuracy between scenarios.

Table 4.2 Evaluation of default remote unlocking model on data preprocessed with packet normalization.

Packet normalization								
category	arrival0	arrival3	arrival5	arrival7	leaving3	leaving5	leaving7	run
Androids	99.4	7.91	16.69	13.37	88.15	96.9	97.26	85.23
iPhones	92.08	10.47	25.1	38.8	73.59	87.52	87.04	84.3

4.2 Simple exponential smoothing

The variation introduced in section 3.6.3 was used to smooth the RSSI data. The results for combinations of initial α_0 and time window are in the table 4.3.

Table 4.3 Evaluation of simple exponential smoothing without normalization.

time window	α_0	model	arrival0	arrival3	arrival5	arrival7	leaving3	leaving5	leaving7	run
Androids										
500ms	0.75	default	98.57	18.18	26.86	20.76	92.42	98.21	98.93	79.55
500ms	0.25	default	98.69	18.18	24.85	19.69	92.54	98.1	98.93	78.7
1000ms	0.75	default	98.69	18.06	26.86	20.53	92.54	98.21	98.93	79.51
1000ms	0.25	default	98.57	18.65	23.67	18.85	92.18	97.98	98.93	78.65
2000ms	0.75	default	98.69	18.18	26.86	20.29	92.65	98.21	98.93	79.51
2000ms	0.25	default	98.57	18.42	22.25	18.62	92.06	97.98	98.69	78.79
iPhones										
500ms	0.75	default	90.59	17.32	32.59	45.93	81.07	92.37	92.68	80.74
500ms	0.25	default	90.59	18.02	32.04	44.94	81.07	91.96	93.66	79.93
1000ms	0.75	default	90.59	17.74	32.59	45.22	80.93	92.65	93.52	80.66
1000ms	0.25	default	90.59	17.88	32.45	45.79	80.23	91.68	94.79	79.58
2000ms	0.75	default	90.72	18.16	32.04	45.22	80.79	92.51	93.66	80.52
2000ms	0.25	default	90.59	18.72	32.45	46.36	78.25	90.71	95.49	78.48

The effects of simple exponential smoothing on models' predictions seem to be quite limited. For both Androids and iPhones, the smoothing changed the prediction accuracy usually by a fraction of a percent or by a few percent. If compared with the default model predictions on unprocessed data, the accuracy in individual scenarios sometimes increases but usually decreases. It is apparent that the overall accuracy got slightly worse if considering all scenarios.

With packet normalization

Another logical step is to add the packet normalization in addition to smoothing. Table 4.4 shows the results.

The results seem to be quite similar to the one with just the packet normalization, further supporting the idea that simple exponential smoothing has only little effect on predictions.

Table 4.4 Evaluation of simple exponential smoothing with normalization.

time window	α_0	model	arrival0	arrival3	arrival5	arrival7	leaving3	leaving5	leaving7	run
Androids										
500ms	0.75	default	99.4	8.03	16.57	13.01	88.27	96.9	97.14	85.12
500ms	0.25	default	99.52	8.03	15.62	12.05	87.56	97.14	96.79	84.66
1000ms	0.75	default	99.4	8.03	16.57	13.01	88.15	96.9	97.14	85.1
1000ms	0.25	default	99.4	8.26	13.73	12.17	86.26	97.14	96.9	84.8
2000ms	0.75	default	99.4	8.03	16.8	13.01	88.15	96.9	97.14	85.14
2000ms	0.25	default	99.4	7.79	12.19	12.05	85.07	97.14	96.79	84.84
iPhones										
500ms	0.75	default	91.71	9.64	25.1	38.8	73.59	87.52	87.46	84.06
500ms	0.25	default	91.58	9.64	23.99	36.95	74.29	87.1	87.61	83.81
1000ms	0.75	default	91.71	10.2	24.27	38.09	73.87	87.52	87.61	84.3
1000ms	0.25	default	91.71	9.36	23.3	35.66	72.88	86.55	89.3	83.26
2000ms	0.75	default	91.83	10.34	23.86	36.95	73.59	87.38	88.17	84.3
2000ms	0.25	default	91.71	9.64	23.3	36.66	70.48	85.3	90.85	82.34

4.3 Simple moving average smoothing

The simple moving average is great at reducing random noise and is often used in positioning applications for RSSI preprocessing. Table 4.5 shows the effects of left-handed simple moving average smoothing with four sizes of time windows.

Table 4.5 Evaluation of simple moving average smoothing without normalization.

time window	model	arrival0	arrival3	arrival5	arrival7	leaving3	leaving5	leaving7	run
Androids									
400ms	default	98.09	18.65	25.56	20.41	92.3	97.98	98.93	78.7
1000ms	default	97.97	19.95	23.08	19.57	90.76	97.74	99.05	78.74
1600ms	default	97.97	19.01	20.83	18.14	87.2	97.86	98.81	77.77
2200ms	default	97.01	18.06	17.51	18.74	81.64	97.98	98.57	75.95
iPhones									
400ms	default	90.97	18.02	32.73	44.94	81.64	92.09	92.54	80.23
1000ms	default	90.84	18.44	32.73	44.94	79.94	91.96	92.96	80.29
1600ms	default	90.84	19.83	32.59	43.79	77.12	90.43	94.37	78.45
2200ms	default	90.1	22.49	34.54	44.37	71.75	88.35	94.37	76.31

The results are disappointing. It seems to do worse than simple exponential smoothing and just makes the accuracy decrease across all scenarios.

With packet normalization

Table 4.6 Evaluation of simple moving average smoothing with normalization.

time window	model	arrival0	arrival3	arrival5	arrival7	leaving3	leaving5	leaving7	run
Androids									
400ms	default	99.64	7.79	15.62	12.53	86.14	97.14	96.9	84.54
1000ms	default	99.28	8.26	15.03	11.58	81.75	96.9	96.79	85.0
1600ms	default	99.04	7.91	13.25	11.34	74.88	96.9	96.9	83.57
2200ms	default	99.16	7.44	10.77	12.17	66.94	96.79	97.26	82.37
iPhones									
400ms	default	91.83	10.2	25.52	38.09	73.87	86.82	86.62	83.42
1000ms	default	91.96	8.94	24.27	37.23	73.73	88.07	87.75	83.93
1600ms	default	91.71	11.31	24.83	35.52	69.49	85.71	90.14	82.22
2200ms	default	91.21	10.61	25.52	36.66	64.27	82.8	90.28	80.29

The packet normalization in combination with a simple moving average, gives similar results

as in combination with simple exponential smoothing.

4.4 Savitzky-Golay smoothing

Let's use the Savitzky-Golay filter. Table 4.7 shows the effects of smoothing with several time windows with a combination of 2-degree or 3-degree polynomials to fit the RSSI.

Table 4.7 Evaluation of Savitzky-Golay smoothing without normalization.

time window	degree	model	arrival0	arrival3	arrival5	arrival7	leaving3	leaving5	leaving7	run
Androids										
500ms	2	default	98.21	19.01	27.93	22.67	91.59	98.45	98.81	77.19
500ms	3	default	98.21	19.01	27.93	22.67	91.59	98.45	98.81	77.19
1000ms	2	default	97.73	19.13	28.05	22.79	91.23	98.33	98.69	76.92
1000ms	3	default	97.73	19.13	28.05	22.79	91.23	98.33	98.69	76.92
2000ms	2	default	98.21	20.66	28.99	23.63	92.06	97.98	99.17	77.45
2000ms	3	default	98.21	20.66	28.99	23.63	92.06	97.98	99.17	77.45
iPhones										
500ms	2	default	90.59	17.6	31.76	45.65	81.5	92.37	92.11	79.56
500ms	3	default	90.59	17.6	31.76	45.65	81.5	92.37	92.11	79.56
1000ms	2	default	91.09	17.32	31.9	45.36	81.64	91.68	91.83	78.5
1000ms	3	default	91.09	17.32	31.9	45.36	81.64	91.68	91.83	78.5
2000ms	2	default	90.47	18.58	33.29	45.36	81.21	92.09	90.99	78.11
2000ms	3	default	90.47	18.58	33.29	45.36	81.21	92.09	90.99	78.11

The effects of smoothing are very weak. The performance barely changes, n scenarios arrival0 or run it gets worse. But in other scenarios, it slightly improves.

With packet normalization

Table 4.8 Evaluation of Savitzky-Golay smoothing with normalization.

time window	degree	model	arrival0	arrival3	arrival5	arrival7	leaving3	leaving5	leaving7	run
Androids										
500ms	2	default	99.52	8.38	17.75	13.48	88.03	96.9	96.67	83.71
500ms	3	default	99.52	8.38	17.75	13.48	88.03	96.9	96.67	83.71
1000ms	2	default	99.52	8.62	18.34	15.04	86.97	97.02	96.9	83.43
1000ms	3	default	99.52	8.62	18.34	15.04	86.97	97.02	96.9	83.43
2000ms	2	default	99.28	9.33	19.17	15.04	86.37	96.55	96.79	83.89
2000ms	3	default	99.28	9.33	19.17	15.04	86.37	96.55	96.79	83.89
iPhones										
500ms	2	default	91.34	10.47	25.1	38.23	73.59	87.52	86.34	83.07
500ms	3	default	91.34	10.47	25.1	38.23	73.59	87.52	86.34	83.07
1000ms	2	default	91.83	11.59	25.66	38.8	74.01	86.96	85.92	82.3
1000ms	3	default	91.83	11.59	25.66	38.8	74.01	86.96	85.92	82.3
2000ms	2	default	91.46	11.17	25.8	37.38	73.31	86.13	85.77	81.52
2000ms	3	default	91.46	11.17	25.8	37.38	73.31	86.13	85.77	81.52

The results are analogical for simple exponential smoothing as simple moving average smoothing.

4.5 Discussion

The results from previous sections are a bit disappointing. On the one hand, it clearly showed that preprocessing RSSI using smoothing and packet normalization, influences algorithm predictions. On the other hand, the effects, especially in the case of smoothing are very questionable.

All of the smoothing methods have little to no effect on algorithm performance. Often making the performance generally worse than in the original version with unprocessed data. This is especially strange because in reviewed papers the preprocessing step always helped to improve the positioning accuracy. There are possible reasons why in this case the smoothing does not help: Either the model is so robust that it can natively filter out random noise in the data. Therefore the smoothing cannot help the algorithm to do any better. The other option is the exact opposite. The algorithm configuration overfits the data and incorporates the noise into its decision process so much that the algorithm is sensitive to any variance in the data. This might explain why in some scenarios the performance dropped after the smoothing. This could have happened due to the presence of non-representative data, which could have compromised the algorithm configuration process. During the data exploration, there were a few indications that this might be the case. But to confirm this, it would be necessary to perform an exhaustive data analysis that falls out of the scope of this thesis.

The packet normalization seems to be just trading-off accuracy in scenarios that should not open the door with the ones that should. The trade-off is apparent in all scenarios. The bright side of this is that there might be a possibility to take advantage of this. If the algorithm would be reconfigured, it could help to achieve better performance in scenarios that should not open the door, while keeping the original opening accuracy in scenarios that should trigger an opening. If this is possible and worthwhile is a question that cannot be answered in this thesis, since it would require extensive tuning of the algorithm and that is not an aim of this thesis.

Regarding model-wise normalization, it seems like it is not an option. According to section 3.4.2, the differences in RSSI in the same scenario between phones are unpredictable and occur even with different instances of the same model. Therefore normalization algorithm could not be proposed and tested.

Conclusion

This thesis introduced key concepts to understand how to use time series smoothing and normalization methods to improve RSSI-based approach detection. Multiple preprocessing techniques were reviewed and implemented to test their potential to improve the existing algorithm.

The theoretical part shows how external factors, like temperature, positioning, or human movement affect RSSI measurements. Unfortunately, it became clear that the RSSI is not an ideal metric for localization, due to its volatile nature. The reviewed methods of smoothing remove outliers from the measurements and highlight the overall trend, which helps to compensate for RSSI fluctuations partially.

The analytical part explored data from 2N Telekomunikace a.s. and managed to get some useful insights. In it were shown the approaches with mobile phones to door-unlocking units in the RSSI time series. That part also discussed how different models/brands affect the measurements. Then a packet analysis was performed to investigate the relationship between advertisement and connection packet and a suitable transformation for each model was proposed. And of course, some of the time series smoothing methods, introduced in the theoretical, part were implemented and their effects visualized.

Experiments found that improving the existing algorithm with preprocessing of RSSI is troublesome. The positive effects of smoothing were negligible and in some cases, the algorithm performed worse with smoothed RSSI than with the original unprocessed one. The effects of normalization were more apparent, but to truly take advantage of them more work has to be done.

In future work, there are many ways of improving the algorithm's performance. Either, there is a possibility of completely switching from RSSI positioning to different technology. For example, UWB (Ultra wide band), which is an emerging RF technology, shows promising results as mentioned in [45]. Another way might be to gather more representative data and try to improve the preprocessing step and algorithm configuration.

Appendix A

Packet correlation tables

Packet correlation - Androids			
phone	mean	median	var
asuszenfone8_1	0.542	0.802	0.174
cat_s61	0.707	0.741	0.028
cubotx20pro_02	0.815	0.835	0.012
honor10	0.676	0.717	0.027
honor50	0.824	0.851	0.011
htcdesire21pro	0.737	0.782	0.026
htc_u11_life	0.674	0.742	0.06
huawei_y7_2019	0.744	0.773	0.011
lg	0.842	0.864	0.008
mobicelr7_02	0.832	0.857	0.01
myphone_prime5_02	0.717	0.733	0.01
nokia	0.758	0.804	0.033
nokia_x20	0.783	0.838	0.037
oneplusnord_02	0.788	0.825	0.018
opporx17_1	0.702	0.738	0.029
oppo_reno3_02	0.79	0.809	0.007
oukitel_2	0.689	0.712	0.018
pixel6	0.565	0.677	0.076
realme	0.788	0.825	0.018
samsungs9	0.758	0.784	0.017
samsung_x	0.841	0.872	0.016
sony	0.781	0.858	0.052
ssa51	0.733	0.832	0.061
tlc20pro	0.743	0.77	0.018
xiaomi_mi_8_lite	0.849	0.881	0.011

Packet correlation - iPhones			
phone	mean	median	var
iphone12_1	0.658	0.75	0.081
iphone12_1221	0.803	0.834	0.018
iphone12_1222	0.615	0.701	0.064
iphone12_v34	0.608	0.672	0.067
iphone12_v35	0.624	0.642	0.045
iphone13_1	0.691	0.723	0.029
iphone13_v34	0.81	0.833	0.014
iphone13_v35	0.632	0.697	0.057
iphone7_v34	0.608	0.655	0.027
iphone7_v35	0.608	0.665	0.04
iphone8_1221	0.54	0.59	0.06
iphone8_1221_2	0.559	0.597	0.05
iphone8_v34	0.634	0.69	0.05
iphone8_v35	0.561	0.631	0.061
iphonese	0.589	0.644	0.064
iphonese2020	0.591	0.636	0.069
iphonese2020_v34	0.467	0.511	0.08
iphonese2020_v35	0.614	0.669	0.057
iphonexr	0.66	0.724	0.061
iphonexr_v34	0.633	0.685	0.062
iphonexr_v35	0.669	0.738	0.058

Bibliography

1. BLUETOOTH®. *Origin of the Name* [online] [https://www.bluetooth.com/about-us/bluetooth-origin/]. (Accessed on 04/10/2023).
2. BLUETOOTH®. *Bluetooth Technology Overview* [online] [https://www.bluetooth.com/learn-about-bluetooth/tech-overview/]. (Accessed on 04/10/2023).
3. BLUETOOTH®. *Bluetooth_Technology_Overview_Graphic.png (1494×1185)* [online] [https://www.bluetooth.com/wp-content/uploads/2021/01/Bluetooth_Technology_Overview_Graphic.png?time=8675309]. (Accessed on 04/10/2023).
4. AFANEH, Mohammad. *(175) Ellisys Bluetooth Video 2: Generic Access Profile - YouTube* [online] [https://www.youtube.com/watch?v=8OfOwD8f2VI]. Ellisys, 2018-04. (Accessed on 04/10/2023).
5. OLIVIA'S PC. *1*7R_hkwuk8v7gClsVXtoqPw.png (1708×824)* [online] [https://miro.medium.com/v2/resize:fit:3416/1*7R_hkwuk8v7gClsVXtoqPw.png]. Medium, 2019-09. (Accessed on 04/10/2023).
6. AFANEH, Mohammad. *Ellisys Bluetooth Video 3: Advertisements - YouTube* [online] [https://www.youtube.com/watch?v=be9ct7OKI7s]. Ellisys, 2018-04. (Accessed on 04/10/2023).
7. SJÖBERG, Katrin; KAREDAL, Johan; MOE, Marie; KRISTIANSEN, Øyvind; SØRÅSEN, Runar; UHLEMANN, Elisabeth; TUFVESSON, Fredrik; EVENSEN, Knut; STRÖM, Erik G. *Measuring and using the RSSI of IEEE 802.11p* [online] [https://www.diva-portal.org/smash/get/diva2:390793/FULLTEXT01.pdf]. (Accessed on 04/10/2023).
8. *Decibel-milliwatt (dBm)* [online] [https://www.rapidtables.com/electric/dBm.html]. RapidTables. (Accessed on 04/10/2023).
9. GAO, Vincent. *Proximity and RSSI — Bluetooth® Technology Website* [online] [https://www.bluetooth.com/blog/proximity-and-rssi/]. Bluetooth®, 2015-09. (Accessed on 04/10/2023).
10. CABRERA-MORA; FLAVIO; XIAO, Jizhong. Preprocessing technique to signal strength data of wireless sensor network for real-time distance estimation. In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 1537–1542. Available from DOI: 10.1109/ROBOT.2008.4543420.
11. GUIDARA, Amir; FERSI, Ghofrane; DERBEL, Faouzi; JEMAA, Maher Ben. Impacts of Temperature and Humidity variations on RSSI in indoor Wireless Sensor Networks. *Procedia Computer Science*. 2018, vol. 126, pp. 1072–1081. ISSN 1877-0509. Available from DOI: https://doi.org/10.1016/j.procs.2018.08.044. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia.

12. BOUSSAD, Yanis; MAHFOUDI, Mohamed Naoufal; LEGOUT, Arnaud; LIZZI, Leonardo; FERRERO, Fabien; DABBOUS, Walid. Evaluating Smartphone Accuracy for RSSI Measurements. *IEEE Transactions on Instrumentation and Measurement*. 2021, vol. 70, pp. 1–12. Available from DOI: 10.1109/TIM.2020.3048776.

13. BEVELACQUA, Peter Joseph. *Wave Polarization and Antenna Polarization* [online] [https://www.antenna-theory.com/basics/polarization.php]. (Accessed on 04/10/2023).

14. GENTNER, Christian; GÜNTHER, Daniel; KINDT, Philipp H. Identifying the BLE Advertising Channel for Reliable Distance Estimation on Smartphones. *IEEE Access*. 2022, vol. 10, pp. 9563–9575. Available from DOI: 10.1109/ACCESS.2022.3140803.

15. BOORANAWONG, Apidet; SENGCHUAI, Kiattisak; JINDAPETCH, Nattha. Implementation and test of an RSSI-based indoor target localization system: Human movement effects on the accuracy. *Measurement*. 2019, vol. 133, pp. 370–382. ISSN 0263-2241. Available from DOI: https://doi.org/10.1016/j.measurement.2018.10.031.

16. ALOMAINY, Akram; HAO, Yang; OWADALLY, Abdus; PARINI, Clive G.; NECHAYEV, Yuri; CONSTANTINOU, Costas C.; HALL, Peter S. Statistical Analysis and Performance Evaluation for On-Body Radio Propagation With Microstrip Patch Antennas. *IEEE Transactions on Antennas and Propagation*. 2007, vol. 55, no. 1, pp. 245–248. Available from DOI: 10.1109/TAP.2006.888462.

17. SMITH, Steven W. *The Scientist and Engineer's Guide to Digital Signal Processing* [http://www.dspguide.com/ch14.htm]. San Diego: California Technical Publishing, 1997.

18. SPOONER, Chad M. *SPTK: The Moving-Average Filter – Cyclostationary Signal Processing* [online] [https://cyclostationary.blog/2021/05/23/sptk-the-moving-average-filter/]. 2021-05. (Accessed on 04/18/2023).

19. FERNANDO, Jason. *Moving Average (MA): Purpose, Uses, Formula, and Examples* [online] [https://www.investopedia.com/terms/m/movingaverage.asp]. Investopedia. (Last modified on 31/03/2023).

20. SMITH, Steven W. *The Scientist and Engineer's Guide to Digital Signal Processing* [http://www.dspguide.com/ch15.htm]. San Diego: California Technical Publishing, 1997.

21. FANG, Min. *Effect of window size in moving average* [online] [https://mins.space/blog/2020-06-29-moving-average-effect-window-size/]. 2020-06. (Accessed on 04/18/2023).

22. DANCKER, Jonte. *A brief introduction to time series smoothing — by Jonte Dancker — Medium* [online] [https://medium.com/@jodancker/a-brief-introduction-to-time-series-smoothing-4f7ed61f78e1]. Medium, 2022-09. (Accessed on 04/18/2023).

23. HYNDMAN, Rob J. *International Encyclopedia of Statistical Science* [https://robjhyndman.com/papers/movingaverage.pdf]. Springer, 2010. (Accessed on 04/18/2023).

24. SMITH, Steven W. *The Scientist and Engineer's Guide to Digital Signal Processing* [http://www.dspguide.com/ch16.htm]. San Diego: California Technical Publishing, 1997.

25. SCHAFER, Ronald W. What Is a Savitzky-Golay Filter? [Lecture Notes]. *IEEE Signal Processing Magazine*. 2011, vol. 28, no. 4, pp. 111–117. Available from DOI: 10.1109/MSP.2011.941097.

26. PANDIA, Keya; RAVINDRAN, Sourabh; COLE, Randy; KOVACS, Gregory; GIOVANGRANDI, Laurent. Motion artifact cancellation to obtain heart sounds from a single chest-worn accelerometer. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 590–593. Available from DOI: 10.1109/ICASSP.2010.5495553.

27. SUHLING, M.; ARIGOVINDAN, M.; HUNZIKER, P.; UNSER, M. Multiresolution moment filters: theory and applications. *IEEE Transactions on Image Processing*. 2004, vol. 13, no. 4, pp. 484–495. Available from DOI: 10.1109/TIP.2003.819859.

28. SCHMID, Michael; RATH, David; DIEBOLD, Ulrike. Why and How Savitzky–Golay Filters Should Be Replaced. *ACS Measurement Science Au*. 2022, vol. 2, no. 2, pp. 185–196. Available from DOI: 10.1021/acsmeasuresciau.1c00054.

29. GARDNER, Everette S. Exponential smoothing: The state of the art—Part II. *International Journal of Forecasting*. 2006, vol. 22, no. 4, pp. 637–666. ISSN 0169-2070. Available from DOI: https://doi.org/10.1016/j.ijforecast.2006.03.005.

30. ABDERREZAK, Laouafi; MOURAD, Mordjaoui; DJALEL, Dib. Very short-term electricity demand forecasting using adaptive exponential smoothing methods. In: *2014 15th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. 2014, pp. 553–557. Available from DOI: 10.1109/STA.2014.7086716.

31. YIN, Jihao; HAN, Bingnan; YU, Wanke. Hyperspectral image target detection based on exponential smoothing method. In: *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2015, pp. 1869–1872. Available from DOI: 10.1109/IGARSS.2015.7326157.

32. OSTERTAGOVA, Eva; OSTERTAG, Oskar. The Simple Exponential Smoothing Model. In: [https://www.researchgate.net/publication/256088917_The_Simple_Exponential_Smoothing_Model]. 2011.

33. GLEN, Stephanie. *Exponential Smoothing: Definition of Simple, Double and Triple* [online] [https://www.statisticshowto.com/exponential-smoothing/]. StatisticsHowTo.com. (Accessed on 04/18/2023).

34. STANLEY, Greg. *Exponential Filter* [online] [https://gregstanleyandassociates.com/whitepapers/FaultDiagnosis/Filtering/Exponential-Filter/exponential-filter.htm]. (Accessed on 04/18/2023).

35. DASH, Sourav. *Smoothing Techniques for time series data* [online] [https://medium.com/@srv96/smoothing-techniques-for-time-series-data-91cccfd008a2#f381]. Medium, 2020-05. (Accessed on 04/18/2023).

36. CHUSYAIRI, Ahmad; RAMADAR, N. S. Pelsri; BAGIO. The use of exponential smoothing method to predict missing service e-report. In: *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*. 2017, pp. 39–44. Available from DOI: 10.1109/ICITISEE.2017.8285535.

37. ABDERREZAK, Laouafi; MOURAD, Mordjaoui; DJALEL, Dib. Very short-term electricity demand forecasting using adaptive exponential smoothing methods. In: *2014 15th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. 2014, pp. 553–557. Available from DOI: 10.1109/STA.2014.7086716.

38. NIST/SEMATECH. *e-Handbook of Statistical Methods*. [N.d.]. Available from DOI: 10.18434/M32189. (Accessed on 04/18/2023).

39. PORTAL, SAP Help. *Triple Exponential Smoothing* [online] [https://help.sap.com/docs/SAP_HANA_PLATFORM/2cfbc5cf2bc14f028cfbe2a2bba60a50/a25b1dee883a4bc4a984bf9496c7a954.html?version=1.0.12&locale=en-US]. (Accessed on 04/18/2023).

40. CHOOSAKSAKUNWIBOON, Shanatip; TERAWONG, Chawin; SUTTISIRIKUL, Suppakorn; ANANTAVRASILP, Isara; THIEMJARUS, Surapa; WISADSUD, Sodsai; KAEMARUNGSI, Kamol. A pre-processing technique for BLE-based indoor localization. In: *Proceedings of the 12th International Convention on Rehabilitation Engineering nd Assistive Technology*. 2018, pp. 241–244. Available also from: https://www.researchgate.net/profile/Isara-Anantavrasilp/publication/352933782_A_Pre-processing_Technique_for_BLE-based_Indoor_Localization/links/60e02786a6fdccb74500a3dd/A-Pre-processing-Technique-for-BLE-based-Indoor-Localization.pdf.

41. JIANYONG, Zhu; HAIYONG, Luo; ZILI, Chen; ZHAOHUI, Li. RSSI based Bluetooth low energy indoor positioning. In: *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2014, pp. 526–533. Available from DOI: 10.1109/IPIN.2014.7275525.

42. ZHU, Haiping; ALSHARARI, Talal. An Improved RSSI-Based Positioning Method Using Sector Transmission Model and Distance Optimization Technique. *International Journal of Distributed Sensor Networks*. 2015, vol. 11, no. 9, p. 587195. Available from DOI: 10.1155/2015/587195.

43. KORANNE, Sandeep. Hierarchical Data Format 5 : HDF5. In: *Handbook of Open Source Tools*. Boston, MA: Springer US, 2011, pp. 191–200. Available from DOI: 10.1007/978-1-4419-7719-9_10.

44. RATNER, Bruce. The correlation coefficient: Its values range between +1/1, or do they? *Journal of Targeting, Measurement and Analysis for Marketing*. 2009, vol. 17, no. 2, pp. 139–142. ISSN 1479-1862. Available from DOI: 10.1057/jt.2009.5.

45. ALARIFI, Abdulrahman; AL-SALMAN, AbdulMalik; ALSALEH, Mansour; ALNAFESSAH, Ahmad; AL-HADHRAMI, Suheer; AL-AMMAR, Mai A.; AL-KHALIFA, Hend S. Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances. *Sensors*. 2016, vol. 16, no. 5. ISSN 1424-8220. Available from DOI: 10.3390/s16050707.

Content of attached medium

- `requirements.txt` — list of used Python packages
- `Analysis` — directory containing some of the analyses
  - `Experiments` — analyses regarding preprocessing evaluation
  - `Packets` — analyses regarding packet normalization
- `Scripts`
  - `Helpers` — auxilary python utilities
    - `__init__.py`
    - `analysis.py` — utilities for data analysis
    - `data_transformation.py` — utilities for data loading and preprocessing
    - `filters.py` — utilities for smoothing RSSI
    - `packet_transformation.py` — utilities for normalizing packets
    - `packet_transformation.py` — utilities for plotting RSSI
  - `Notebooks` — Jupyter notebooks
    - `data_analysis.ipynb` — data analysis
    - `data_exploration.ipynb` — basic data exploration
    - `packet_difference.ipynb` — packet normalization
    - `time_series_smoothing.ipynb` — RSSI smoothing
  - `setup.py`
- `thesis_source` — source code for thesis.pdf in LaTeX
- `thesis.pdf` — thesis text in .pdf format