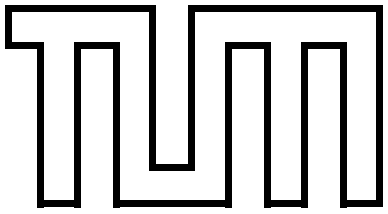


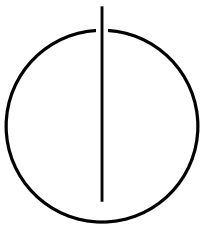
TECHNISCHE UNIVERSITÄT MÜNCHEN

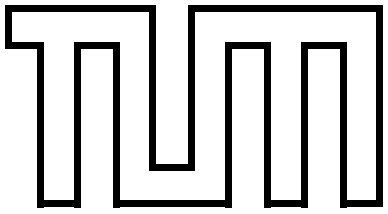
DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Passive WiFi-based Indoor Localization With Semantic Correction

Salma Farag





TECHNISCHE UNIVERSITÄT MÜNCHEN

DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Passive WiFi-based Indoor Localization With Semantic Correction

Passive WiFi-basierte Indoor-Lokalisierung mit semantischer Korrektur

Author:	Salma Farag
Supervisor:	Prof. Dr. Dr. h.c. Manfred Broy
Advisor 1:	Dipl.-Ing. Georgios Pipelidis
Advisor 2:	Dipl.-Ing. Nikolaos Tsiamitros
Submission Date:	03.12.2019

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 03.12.2019 Salma Farag

Acknowledgments

First off, I am thankful to my advisors Georgios Pipelidis and Nikolaos Tsiamitros for allowing me to work on and learn about this extremely interesting topic. Your guidance and support have been incredible, and despite our differences, I could not have asked for better advisors.

I wish to thank my team at Ariadne Maps, especially Nam, for helping me set up experiments and being the wonderful person that he is.

I am especially grateful to my best friend back home, Salloma, for being there with me every step of the way and my friends Irene and Polina for taking such good care of me and being my family in Munich. I am grateful to my friends Shwetha and Ninah for the good times, Faisal, for his endless wisdom, Muawiya, for all his advice, and Ahmad for proofreading my thesis.

Last but not least, I would like to express my sincerest gratitude to my family, who, without their love and support, this thesis would not have been possible.

Abstract

With the growing proliferation of smartphones and the diminishing cost of sensors and high-end devices, there are growing demands for LBSs (Location-based services) in commercial and governmental infrastructures. This thesis explores the methods of non-invasively localizing phones indoors, where people are now spending the majority of their time. We talk about why GPS (Global Positioning Systems) are not used indoors and how Wi-Fi is a more viable option due to its already ubiquitous nature. We implement physical localization using WNLS (Weighted Non-linear Least Squares) and are able to achieve up to a baseline of 1.61 m median resolution. Furthermore, we perform semantic localization through polygon projection, which improves our overall median localization accuracy to 1.38 m. We implement an approach that computes a responsive uncertainty radius through RSSI (received signal strength indicator) and the user's velocity. In addition, we implement and compare the performance of a supervised and an unsupervised HMM (hidden Markov model) in the context of semantic localization correction. Lastly, we implement a low-effort, room-based fingerprinting technique and achieve a cross-validation score of over 91% without any IMU data collection.

Contents

List of Abbreviations

AP	Access Point
CDF	Cumulative Distribution Function
CSI	Channel State Information
DLoS	Direct Line of Sight
EM	Expectation-maximization
FSPL	Free-space Path Loss
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HMM	Hidden Markov Model
IMU	Inertial Measurement Unit
IoT	Internet of Things
*k*NN	*k*-Nearest Neighbors
LBS	Location-based Service
MAE	Mean Average Error
MAP	Maximum A Posteriori
MLE	Maximum Likelihood Estimation
MQTT	Message Queuing Telemetry Transport
NLLS	Non-linear Least Squares
NLoS	None Line of Sight
RF	Radio Frequency
RMSE	Root Mean Square Error
RSSI	Received Signal Strength Indicator
ToA	Time-of-Arrival
WNLS	Weighted Non-linear Least Squares

List of Figures

List of Tables

Listings

List of Algorithms

1. Introduction

1.1. Motivation

Global positioning systems (GPS) are more popular today than ever before. Nevertheless, people are spending approximately 90% of their time indoors [1], where GPS signals cannot thoroughly penetrate. Subsequently, this saw a growing demand for indoor location-based services (LBSs), especially since it is estimated that by 2020, each person will have 6.58 network-connected devices [2].

Our phones are constantly emitting a range of radio frequency (RF) signals in the form of cellular data, Wi-Fi, and Bluetooth signals, all of which can be used to estimate our locations to a much higher precision than GPS. Indoor localization relies on these signals to enable its users to navigate their way inside buildings using mobile applications. Not only individuals can benefit from indoor localization but organizations as well, through passive localization. Passive indoor localization does not require a person to download an application on their phone. It only relies on the RF signals which phones are constantly emitting when scanning for APs. This, of course, can be used maliciously to track unsuspecting users. However, when regulated, this technology can have a variety of beneficial applications. For one, it can help businesses and governments improve the quality of their services, as well as improve their cost and efficiency by dynamically adjusting them to the number of customers available, without the need for the customers to download any special software on their phones. It can also give transportation companies insight into the number of passengers aboard, which can help them organize vehicles as needed, cutting on fuel and emissions. Moreover, offices can use this technology to boost their productivity as well as their employees' satisfaction.

Passive localization does not only provide economic benefits for companies and businesses but can be used in other domains as well. For instance, in emergencies and rescue operations, passive localization can help find survivors through their phones. It can also be used to track elderly patients with memory-related diseases (e.g., Alzheimer's disease), allowing them to walk freely in their care facilities. It may also be utilized in homes to help reduce energy consumption by turning off lights, heating, and electronic devices in uninhibited rooms.

1.2. Objective

In this thesis, we investigate ways of how users can be passively localized through their phones using Wi-Fi access points (APs), which are ubiquitous today more than ever. We aim to demonstrate the extent to which users can be passively localized, using minimal knowledge of the user's device and minimal configuration effort. By *passively*, we mean tracking phones using the signals they are constantly broadcasting, without the need for downloading any software.

Moreover, we aim to show the significance of polygon projection in improving not only the semantic accuracy but also the physical accuracy as well. We further demonstrate the effectiveness of semantic correction through hidden Markov models (HMMs) to minimize localization fluctuation and uncover knowledge from noisy data. Lastly, we explore the strengths and weaknesses of different localization configurations through a series of experiments.

1.3. Overview

In Chapter 2, we will look into some relevant background concepts that may be essential for the reader to understand this thesis. In Chapter 3, we describe our methodology, starting with the distance estimation for true range multilateration. We then explore different multilateration implementations to obtain a baseline. In addition, we show how we enhance the baseline's performance by semantic localization through indoor map segmentation and polygon projection. We describe a method to quantify localization uncertainty, which we visualize as the radius of the circle surrounding a localization prediction. Moreover, we implement semantic localization correction using HMMs. We also implement room-based fingerprinting and investigate some of the challenges, including device heterogeneity for both mobile phones and Wi-Fi APs alike. In Chapter 4, we conduct several experiments to evaluate the performance of every component. We study how the system performs under each of these different configurations. In Chapter 5, we look into some of the related work and relevant literature. Finally, in Chapter 6, we discuss our plans for future work.

1.4. Contributions

The following list of contributions have been made in this thesis:

- A true range multilateration implementation using weighted non-linear least squares (WNLS).

- A responsive uncertainty radius that reflects the possible amount of localization error using information from the current, as well as the historical data.
- Semantic localization using indoor map segmentation and nearest polygon projection.
- Semantic correction through supervised and unsupervised HMMs.
- Low-effort, room-based fingerprinting through a trained machine-learning classifier.

2. Background

2.1. Global Navigation Satellite Systems

Global Navigation Satellite System (GNSS), also commonly known as GPS, is the most prominent form of localization today. As the name suggests, GNSS uses satellites orbiting the Earth to locate objects in 3-D using geometry-based techniques. There are at least 24 operational satellites at any given time, orbiting at a distance of approximately 26,600 km in what is known as a GPS Satellite Constellation [3]. The satellites are positioned in such a way that four satellites are covering each of six orbital planes. There is also an extra satellite in case one of the main four stopped working. The number of satellites in every plane is not entirely random, but rather is the minimum number of satellites required to locate an object in 3-D using geometry-based localization techniques (e.g., triangulation). GPS is highly effective for outdoor localization. However, the signals cannot penetrate the indoor environment as effectively, thus are not typically used for indoor localization. This is the motivation behind indoor localization.

2.2. Radio Frequency

Almost all of our everyday wireless communication use channels in the RF range. RF signals are electromagnetic waves that travel in space at the speed of light. Like all waves, they experience reflection, refraction, diffraction, and interference. Therefore, it is essential to have a direct line-of-sight (DLoS) between the transmitter and the receiver for optimal signal transmission. Otherwise, signals experience what is known as multi-path propagation, which means that the transmitted signal reaches the receiver from different directions with varying time delays. This occurs due to the signal reflecting off of intermediate surfaces (e.g., walls or tables) before arriving at the receiver. So when it comes to indoor localization, multi-path propagation can be a significant source of error that needs to be leveraged.

2.2.1. Received Signal Strength Indicator

Another important property of RF signals is that they tend to attenuate at a fairly predictable rate, the further away as they travel from the source. This property enables us to estimate the distance between the transmitter and the receiver with good precision. Wireless devices typically emit Wi-Fi probe requests at an inconsistent rate. However, they can emit at a rate of up to once every 5-6 seconds. Each probe request contains a received signal strength indicator (RSSI) field. As the name suggests, the RSSI is a value that indicates how strong a received signal is, and is measured in decibel-milliwatt (dBm). To be able to estimate the distance from the RSSI, it is essential to understand the RSSI distribution of the devices. Kaemarungs [4] studied this problem and concluded that the RSSI range and distribution vary from one vendor to another. However, the RSSI values generally fall somewhere in the range between 0 and -100 dBm, following a semi-symmetric log-normal distribution that is skewed to the left or the right, based on the frequency ratio of stronger to weaker signals. Figure 2.1 depicts the RSSI distribution constructed from real data, as collected by our Wi-Fi APs in the localization environment over the course of an hour.

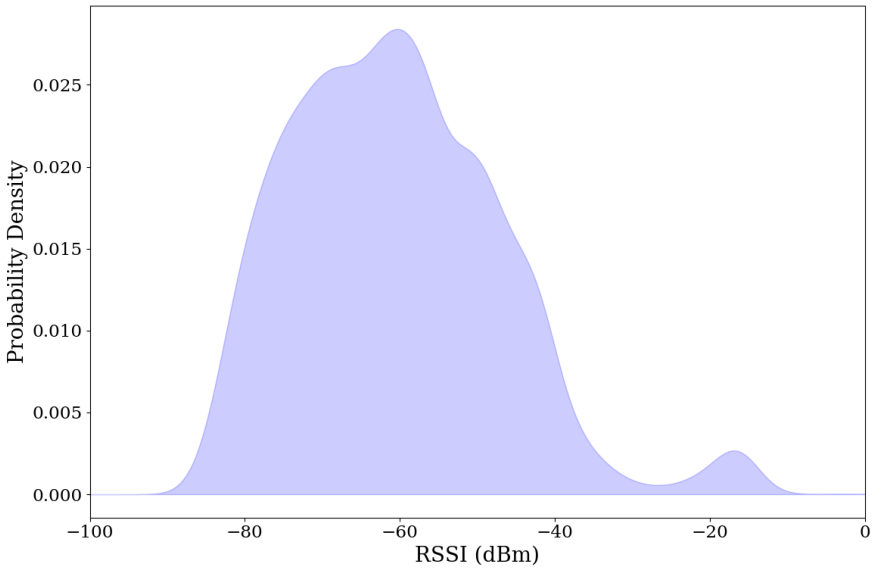


Figure (2.1) The RSSI distribution generated by the data collected from 30 Wi-Fi APs over the duration of an hour.

There already exist different signal propagation models that can help infer the distance from a source using the RSSI value. The most popular is Free-space Path Loss (FSPL) propagation model, which is derived from Friis' transmission formula [5].

2.2.2. Free-space Path Loss

The FSPL model can be mathematically described as:

$$FSPL = \left(\frac{4\pi d}{\lambda}\right)^2, \tag{2.1}$$

such that d is the distance between the transmitter and the receiver, and λ is the transmission signal wavelength. To obtain the FSPL (dBm), λ is replaced by c/f, where c is the speed of light, and f is the frequency (Hz). Taking the common log of the term multiplied by 10, we obtain the following equation:

$$FSPL = 10\log_{10}\left(\left(\frac{4\pi df}{c}\right)^2\right), \tag{2.2}$$

Taking down the power and expanding the log produces:

$$FSPL = 20\log_{10}(d) + 20\log_{10}(f) + 20\log_{10}\left(\frac{4\pi}{c}\right), \tag{2.3}$$

where the last term can be substituted by the empirical constant -27.55 if d is expressed in meters and f is expressed in megahertz. Additionally, we substitute f by 2400 MHz since we are listening for probe requests on the 2.4 GHz frequency band as such:

$$FSPL = 20\log_{10}(d) + 20\log_{10}(2400) - 27.55. \tag{2.4}$$

Finally, to acquire the distance, we move the first term to the left hand side:

$$20\log_{10}(d) = FSPL - 20\log_{10}(2400) + 27.55, \tag{2.5}$$

and we raise the logarithmic base 10 to the power of the terms on the right hand side to obtain:

$$d = 10^{\frac{FSPL - 20\log_{10}(2400) + 27.55}{20}}. \tag{2.6}$$

Now the FSPL can be substituted by the absolute RSSI value, and the distance between the transmitter and receiver can be estimated. Note that the FSPL model assumes a DLoS, and so, it is not suitable to use indoors. However, it is the first step towards understanding propagation models.

2.3. Trilateration

Trilateration is a mathematical technique based on a simple idea: if we want to know the location of an object, which we will call p, on a 2-D grid and we already know that the distance between p and another object a is d_a, then we can say that p lies on the perimeter of a circle such that the center of the circle lies on a, and the radius of the circle is d_a. This allows us to narrow down the location of p to an infinite number of possible locations

that all lie on the perimeter of the circle. If we now know that the distance between p and a is d_a and that from a second object, b is d_b, then we can further narrow down the location of p to exactly the two points where the two circles intersect. To narrow down the location to exactly one, we need a third object c. If we know the distance d_c between p and c, we then have three circles that should intersect in exactly one point, which is the location of p. This scenario can be mathematically formulated as the following system of equations:

$$\begin{aligned}(x_p - x_a)^2 + (y_p - y_a)^2 &= {d_a}^2 \\ (x_p - x_b)^2 + (y_p - y_b)^2 &= {d_b}^2 \\ (x_p - x_c)^2 + (y_p - y_c)^2 &= {d_c}^2,\end{aligned} \tag{2.7}$$

which can be solved simultaneously. It is further illustrated in Figure 2.2. Trilateration is extended and used by GNSS to locate objects in 3-D space using four objects of reference instead of just three.

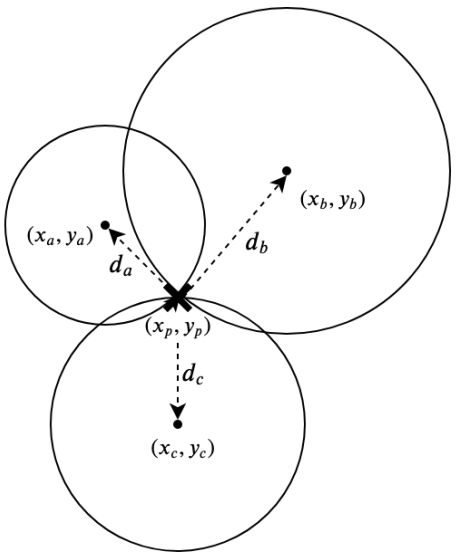


Figure (2.2) Trilateration.

2.4. Hidden Markov Models

Generally speaking, a Markov model is a probabilistic temporal model that describes a Markov process in terms of discrete states. A Markov process is a process that satisfies the Markov assumption, which states that the current state only depends on a finite fixed number of previous states [6]. A hidden Markov model (HMM) is a Markov model with hidden states and one that describes a Markov process using a single random variable. The hidden state of such a variable cannot be directly observed; however, it can be

indirectly inferred from the observations. HMMs are used in various areas, including natural language processing (NLP), speech recognition, and robotics.

An HMM has a set of hidden states L such that:

$$L = \{l_1, l_2, l_3, ..., l_n\}, \tag{2.8}$$

where n is the number of states. In addition to the hidden states, an HMM has a transition model A, where A is the transition probability matrix of size $n \times n$. A_{ij} denotes the probability of transitioning from state l_i to state l_j and can be mathematically expressed as follows:

$$A_{ij} = P(l_j \mid l_i). \tag{2.9}$$

An HMM also has a set of possible observations O such that:

$$O = \{o_1, o_2, o_3, ..., o_m\}, \tag{2.10}$$

where m is the total number of possible observations. Moreover, it has a sensor (sometimes called the observation or the *emission*) model B, where B is a diagonal matrix of size $n \times n$. The emission matrix contains the posterior probabilities of each observation o_t occurring at time t, given we are in a particular state l_t. Hence, an emission probability at time t can be articulated as follows:

$$B_t = P(o_t \mid l_t). \tag{2.11}$$

Lastly, we define π as the probability distribution of starting in each state. It can be expressed as a vector of length n. The prior probability distribution is based on our beliefs before the data is seen. The HMM notation is summarized in Table 2.1.

L	The set of hidden states (locations) in the localization space of size n: $L = \{l_1, l_2, l_3, ..., l_n\}$.
O	The set of possible localization observations of size m: $O = \{o_1, o_2, o_3, ..., o_m\}$.
π	The prior probability distribution vector of size n: $\pi = [\pi_1, \pi_2, \pi_3, ..., \pi_n]$.
A	The transition probability distribution matrix of size $n \times n$.
B	The emission probability distribution matrix of size $n \times n$.

Table (2.1) *The HMM notation.*

Since we are performing localization, our set of states L can be defined as the set of discrete locations within a localization space. The start matrix π should contain the

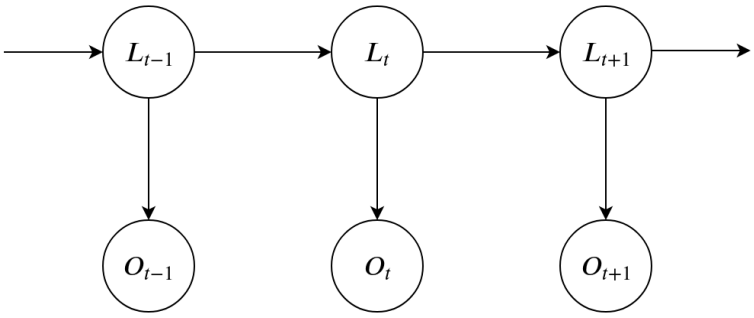


Figure (2.3) A Bayesian network structure of a HMM with the transition model $P(L_t \mid L_{t-1})$ and the observation model is $P(O_t \mid L_t)$.

probability of starting in each location. Moreover, our transition matrix A should contain the probability distribution of transitioning from each location to the other. Finally, our emission matrix B should contain the probability distribution of being observed in each location given the actual location. Figure 2.3 illustrates an HMM expressed as a Bayesian network.

HMMs are used for probabilistic inference in the following tasks:

- **Filtering:** estimating the current state given all previous observations (i.e., forward algorithm).
- **Prediction:** predicting the state at time $t + k$ where $k > 0$, given all previous observations.
- **Smoothing:** computing the posterior probability of being in a former state, given all observations made until the present (i.e., forward-backward algorithm).
- **Most likely explanation:** generating the sequence of states which best explains a sequence of observations (i.e., Viterbi algorithm).

Besides inference, HMMs can be trained to learn the transition and observation models from the data. This can be done in a supervised manner using labeled data, or in an unsupervised manner through the Baum-Welch algorithm or the Viterbi training. Before we jump into the different kinds of training, we first go over some concepts that the reader should know beforehand: the Viterbi algorithm and parameter estimation.

2.4.1. Viterbi Algorithm

The Viterbi algorithm is mainly used in inference for finding the most likely explanation. However, it can be incorporated into unsupervised training, or decoding (which is the same as inference). The algorithm works by maximizing the probability of the sequence

of hidden states given a sequence of observations. This problem can be viewed as a graph search through the possible states as illustrated in Figure 2.4, and can be mathematically expressed as follows:

$$\max_{l_1 \ldots l_t} \Big(P(l_1, ..., l_t, L_{t+1} \mid o_{1:t+1}) \Big), \tag{2.12}$$

which using Bayes' theorem can be rewritten as:

$$\alpha P(o_{t+1} \mid L_{t+1}) \max_{l_t} \left(P(L_{t+1} \mid l_t) \max_{l_1 \ldots l_{t-1}} \Big(P(l_1, ..., l_{t-1}, l_t \mid o_{1:t}) \Big) \right), \tag{2.13}$$

where α is the normalization factor. In the context of localization, given a phone's sequence of localization predictions, we can try to find the most likely path a person has taken.

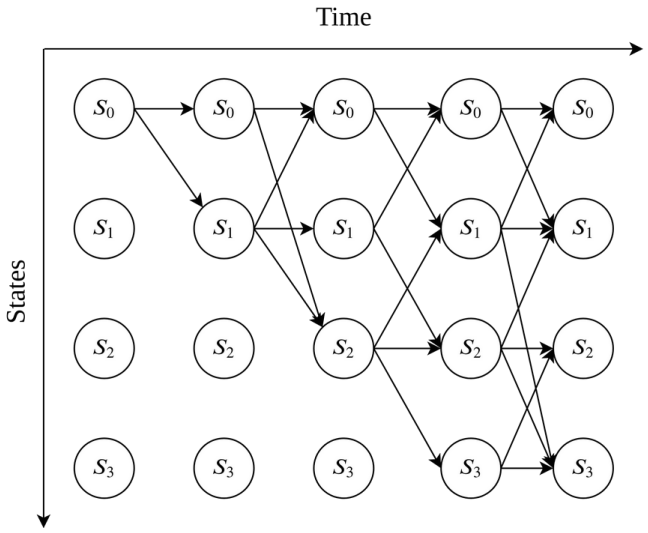


Figure (2.4) Viterbi algorithm.

2.4.2. Parameter Estimation

For an HMM, the parameters we want to learn from the data are the transition and emission probability distributions. There are a few ways to estimate those parameters. In this thesis, we focus on two estimators: the maximum likelihood estimation (MLE) and the maximum a posteriori (MAP).

2.4.2.1. Maximum Likelihood Estimation

Given a set of parameters θ, such that $\theta = \{\pi, A, B\}$, the MLE tries to maximize the likelihood of the data. For computational simplicity, the log-likelihood is specifically

used instead. For instance, to learn the transition model, the number of transitions from each state l_i to every other state is counted, and θ's new values are set such that the log-likelihood of the transition frequency in the data is maximized:

$$\theta_{MLE} = \arg\max_{\theta} \log P(D \mid \theta). \tag{2.14}$$

This is similarly done for the observation model.

MLE assumes that the data is independent and identically distributed (i.i.d.). This contradicts the Markov assumption, which states that each state depends on a certain number of past states. Moreover, if the training data is limited, it becomes very easy to overfit the model to the data. This is especially true if the data is missing a possible transition from a state to another. The model will subsequently not learn such a transition. Thus, MLE may not be the best parameter estimation method for HMM training. A prior distribution, therefore, needs to be defined to ensure no missing transitions or emissions, thereby reducing the chance of overfitting.

2.4.2.2. Maximum A Posteriori

MAP is another way to estimate the parameters of a model during learning. It does so by maximizing the posterior probability of the model parameters θ given the data D:

$$\theta_{MAP} = \arg\max_{\theta} P(\theta \mid D), \tag{2.15}$$

which can be expanded using Bayes' theorem to:

$$\theta_{MAP} = \arg\max_{\theta} P(\theta) P(D \mid \theta). \tag{2.16}$$

This allows us to set a prior distribution $P(\theta)$, making it more difficult to overfit the model to the data.

2.4.3. Supervised Learning

To train a supervised HMM, MAP is usually used to maximize the probability of θ given the labeled data. Naturally, the model is fed an initial probability distribution for π, A, and B. The parameters are refined in several iterations until convergence.

2.4.4. Unsupervised Learning

For unsupervised learning, the Baum-Welch algorithm is generally preferred. However, the Viterbi training is also a possibility. We will explain both and why the Baum-Welch is generally better.

2.4.4.1. Viterbi Training

Viterbi training implements the Viterbi algorithm to learn the transition and emission probabilities of an HMM in an unsupervised fashion. It works as follows:

Algorithm 1 Viterbi Training

1: Set initial values of θ to arbitrary values
2: **while** not converged **do**
3: Run Viterbi algorithm to find best path p^*
4: Update θ to maximize the likelihood of p^* as well as the start, transition and emission frequencies in the data
5: **end while**

The Viterbi training only uses information from the best path p^*. This has proven to be not so effective in training an HMM since there is other relevant information that can be learned from all the other paths. Thus, the Viterbi training generally does not perform so well.

2.4.4.2. Baum-Welch Algorithm

While the Viterbi algorithm computes the probability of the most likely path, the Baum-Welch algorithm computes the probability over all paths. It does so by implementing expectation-maximization (EM), which uses the forward-backward algorithm in every iteration to compute and improve the distribution of the past states given the observations made up to the present time, i.e., $P(L_k \mid o_{1:t})$, where $0 \leq k < t$.

- **Expectation-maximization:** the EM algorithm consists of two steps:

  - **E-step**: the algorithm first "expects" or pretends to know the parameter values of the model.

  - **M-step**: the algorithm then tries to maximize the log-likelihood of the parameters given the data using MLE or MAP.

  On the first iteration, the parameters are initialized with guesses or random values. The algorithm then iteratively executes these two steps until it converges to a local optimum. EM can be summarized in the following equation:

$$\theta^{(i+1)} = \arg\max_{\theta} \sum_{z} P(Z = z \mid x, \theta^{(i)}) L(x, Z = z \mid \theta), \tag{2.17}$$

  where Z denotes all the hidden variables, x denotes all the observed values, and θ denotes all the model probability parameters.

- **Forward-backward Algorithm:** the forward-backward algorithm in itself involves two steps: a forward (filtering) step and a backward step. In the forward step, the algorithm finds the most likely state at every time step summed over all the paths that lead to this state, weighted by the probability of every path. In the backward step, the algorithm traverses backward through all the paths, finding the most likely state at every time step, summed over all the paths up to this state, weighted by the probability of every path. Using Bayes' theorem, the forward-backward algorithm can be formulated as follows:

$$\begin{aligned} P(L_k \mid o_{1:t}) &= P(L_k \mid o_{1:k}, o_{k+1:t}) \\ &= \alpha P(L_k \mid o_{1:k}) P(o_{k+1:t} \mid L_k, o_{1:k}) \\ &= \alpha P(L_k \mid o_{1:k}) P(o_{k+1:t} \mid L_k) \\ &= \alpha f_{1:k} \times b_{k+1:t}, \end{aligned} \tag{2.18}$$

  where $f_{1:k}$ is the forward step and $b_{k+1:t}$ is the backward step.

  The Baum-Welch algorithm can be summed up as follows:

Algorithm 2 Baum-Welch Algorithm

```
Set initial values of θ to arbitrary values
while not converged do
    Run forward algorithm
    Run backward algorithm
    E-step: compute new log-likelihood P(D | θ)
    M-step: update θ to values that maximize the likelihood of the data
end while
```

2.4.5. Decoding

After the model is trained, it should be able to decode a sequence of observations into the most likely sequence of states that produced it. There are two ways this can be done:

- **Viterbi decoding:** uses the Viterbi algorithm to find the most likely sequence of states.

- **Posterior decoding:** also known as MAP decoding, uses the forward-backward algorithm to produce a path containing the most likely state at every time step, which led to the observations.

2.5. Fingerprinting

The main idea behind fingerprinting is that different locations in a localization space have different but unique RSSI fingerprints (Figure 2.5). By collecting RSSI data in these different locations, one can construct a radio map of the localization space that can be used to infer a phone's location, given its RSSI fingerprint. There are two ways to implement fingerprinting: a *probabilistic* method and a *machine learning* method.

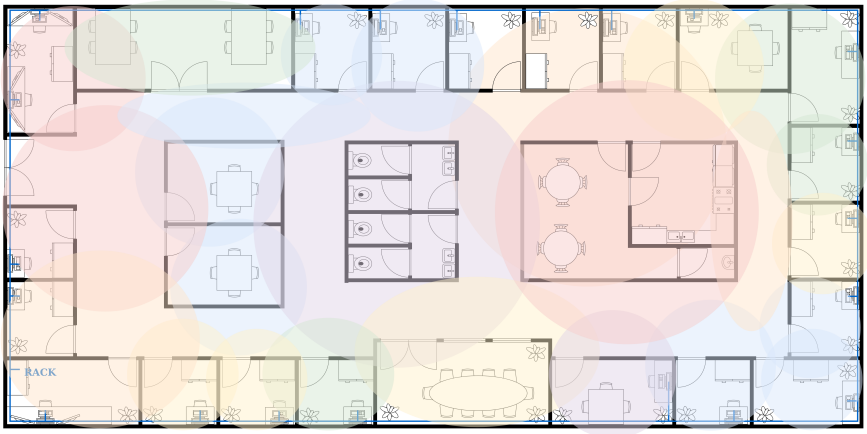


Figure (2.5) A simple illustration of varying Wi-Fi signal fingerprints in an office floor plan. Similar colors do not necessarily indicate the same fingerprint. The fingerprints are influenced by, but not restricted to the floor topology.

2.5.1. The Probabilistic Method

This method involves two phases: an offline phase and an online phase. During the offline phase, a surveyor surveys the area with a phone to collect RSSI data. In the online phase, Bayes' theorem is used to compute the posterior probability $P(l_i \mid s)$, which is the probability of a phone being in location l_i, where l_i is one of n possible locations, namely $l_1, l_2, l_3, ..., l_n$, given a vector s of RSSI values detected by the nearby APs. The goal is to find the location l_i, which maximizes the likelihood of the vector s:

$$\arg\max_i [P(l_i \mid s)]. \tag{2.19}$$

2.5.2. The Machine Learning Method

Similar to the probabilistic approach, a classic machine-learning-based fingerprinting involves two phases: an offline phase and an online phase. In the offline phase, a

surveyor also surveys the area with a phone to collect the data needed to train a classifier (e.g., *k*NN, random forest, or neural network) to recognize different fingerprints. In the online phase, the classifier matches a vector of RSSI with the most similar fingerprint, and the location of the phone can be predicted [7]. Typically, the fingerprinting makes use of patterns encountered in the IMU data to detect and count steps to estimate the distance and direction in which the user is moving, hence make a more informed prediction about their location. In the case of passive localization, however, such data is inaccessible.

2.6. Evaluation Metrics

There are several ways to evaluate the performance of a localization system. Some of the evaluation criteria [8] which are taken into consideration are:

1. **Scalability**: evaluates how well the system can accommodate a large-scale localization project, which may involve hundreds or thousands of APs and users. Moreover, it evaluates the centrality of the system and the ease of extending the algorithm to distributed systems.

2. **Accuracy**: indicates how close the predicted location is to the real location, usually measured in meters. The smaller the distance between the two, the higher the accuracy.

3. **Resilience to error and noise**: measures the robustness of the system to noise and its ability to mitigate it.

4. **Coverage**: evaluates the physical area covered by the localization APs and includes:

    - Density: the mean number of APs per a fixed area (e.g., 4 m^2).

    - Anchor placement: computed through the geometric dilution of precision (GDoP) [9] metric, whose value indicates the quality of the nodes' geometric distribution. Generally, the nodes should be distributed in such a way that maximizes the area between them.

5. **Cost**: how expensive it is to implement the algorithm in terms of hardware, software, and power consumption.

In this thesis, we focus on evaluating the accuracy while maintaining a scalable and resilient system, without any special hardware requirements.

There are two popular metrics for evaluating physical localization accuracy:

- **Mean absolute error (MAE)**: evaluates the overall average distance between the true and predicted locations without accounting for the direction. All errors are given the same weight, and the formula is described as follows:

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|x_j - \hat{x}_j|, \tag{2.20}$$

  where x_j is a true location and $\hat{x}_j$ is its corresponding prediction.

- **Root mean squared error (RMSE)**: measures the overall average error in terms of the square root of the mean squared error. Since the error is squared, larger errors are penalized more than smaller errors. The formula is expressed as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_j - \hat{x}_j)^2}. \tag{2.21}$$

The way each of the errors is computed guarantees that the $RMSE \geq MAE$. Since $RMSE$ is more sensitive to anomalies, MAE is generally preferred over $RMSE$.

3. Methodology

In this chapter, we explain our approach in detail. We begin by giving an overview of the system, and then taking a closer look at every component.

3.1. System Architecture

Figure 3.1 shows an overview of our system components and how they relate to one another. There are two main possibilities: localization through multilateration or through fingerprinting. In multilateration, RSSI data is obtained from Wi-Fi APs and is used to compute the physical location of the phone through a series of steps. The uncertainty is then quantified before semantic correction is performed. In fingerprinting, RSSI data is used to train a classifier to perform room-based localization, and the predictions can also be fed into an HMM for correction. Either of the two, or an ensemble can be used.

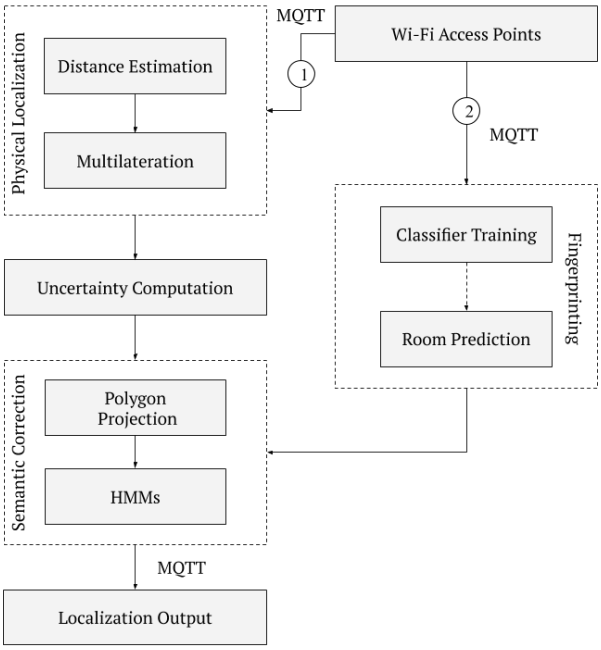


Figure (3.1) System architecture.

3.2. Wi-Fi Access Points

Our first component is the Wi-Fi APs, which have been placed around the corridor and in some of the rooms of the Software & Systems Engineering chair at the faculty of Informatics at the Technical University in Munich. The area extends over approximately 400 m², including restricted area (e.g., personnel offices) with a 30 m long corridor, and offices with an average area of 16 m². We placed 29 APs whose locations were relatively dictated by the positions of power plugs, which fortunately for us were abundant around the chair. The APs were scattered in a way that maximizes their coverage of the desired area. To measure the locations of the APs, we eliminated the vertical component and assumed a 2-D plane for simplicity. We chose one of the APs to be the anchor node to which we assigned the location $(0, 0)$. We then manually measured the x and y components of each of the other APs to the anchor node in meters and recorded them as the APs locations. This procedure needed to be done only once as the APs were stationary. All that was left to know was the geographic location of the anchor node and a way to convert the locations relative to it into latitude and longitude. The prior was easily obtained from OpenStreetMap [10]. For the latter, we assumed a flat Earth, which is a reasonable approximation of the Earth's curvature over relatively small areas, like buildings. We measured the difference in the longitude and latitude of the Garching-Forschungszentrum subway station on the map, which is very close to our localization area. We used the computed values to convert an object's relative location to the anchor to their geographic location. The deployed APs physical distribution is roughly illustrated in Figure 3.2.

For the APs, we used LogiLink WL0151 antennas attached to Raspberry Pi 3 Model B+ devices. The APs functioned as receivers, detecting Wi-Fi probes emitted in their proximity within the 2.4 GHz frequency range. For the experiments conducted and the data collected during the course of the thesis, five different smartphones were used interchangeably: two SM-G928F, an SM-J106H, an SM-G925F, and an LG-D855. The smartphones were borrowed from members of the chair based on availability, excluding those that do not probe often, and those that perform MAC address randomization. For convenience, we will refer to the phones as phones A through E as per Table 3.1.

Name	Commercial Name	Model
A	Samsung Galaxy S6 edge	SM-G925F
B	Samsung Galaxy J1 mini prime	SM-J106H
C	Samsung Galaxy S6 edge+	SM-G928F
D	Samsung Galaxy S6 edge+	SM-G928F
E	LG G3	LG-D855

Table (3.1) *The phones used throughout the thesis*

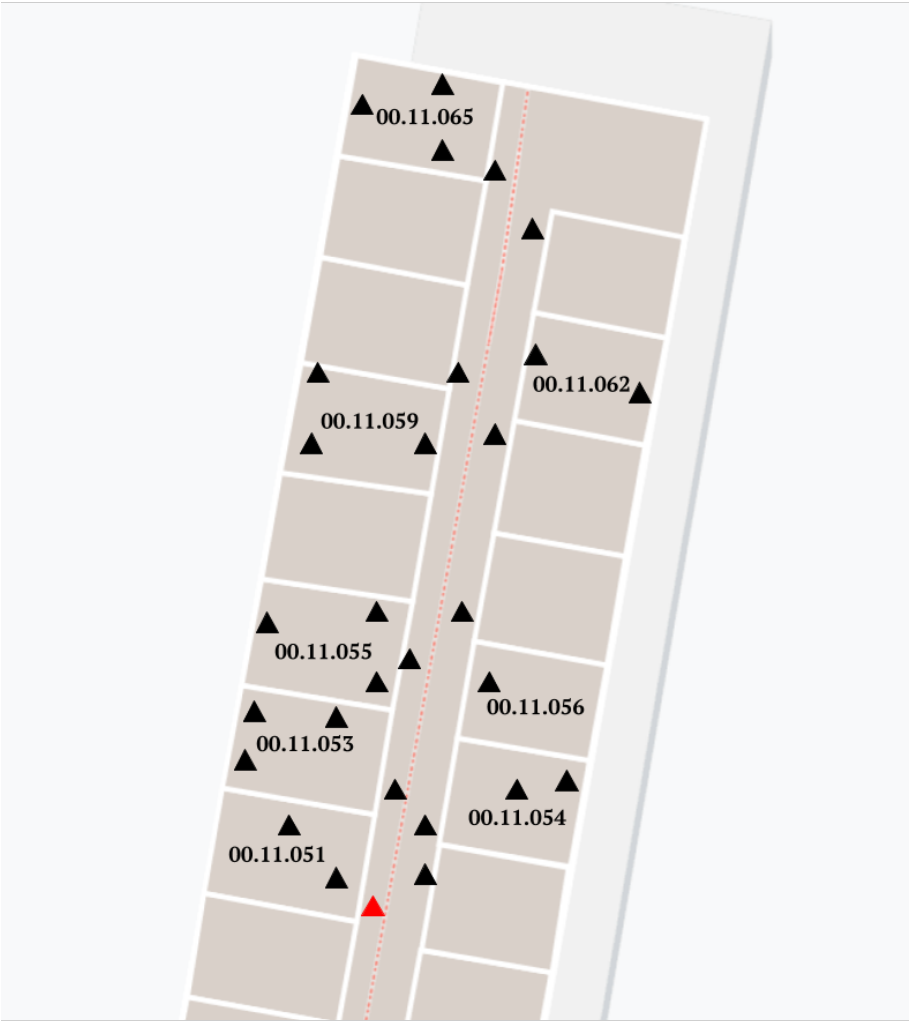


Figure (3.2) The AP distribution in the localization area. The red AP is the anchor/reference node, from which all the other AP relative distances were measured and their relative locations determined. Unlabeled rooms are restricted and were excluded from the localization domain.

To utilize the RSSI data, the APs streamed the data to an MQTT [11] topic to which the localization components were subscribed. The MQTT protocol is present in a variety of applications but is perhaps most prevalent in IoT applications because of its efficiency and simplicity. One can subscribe to different topics to which sensors publish their measurements with minimal delay. We used AWS IoT [12] to implement the MQTT protocol. In our application, a single MQTT message consisted of a MAC address of a detected device and its RSSI along with other data, as shown in the sample JSON array below. The data was also pushed to a Firebase NoSQL database in the same format for visualization. The visualization process itself is out of the scope of this thesis.

Listing (3.1) A JSON array containing three MQTT messages as streamed by the APs to the MQTT topic

```
[
    {
        'sensor_id': 28,
        'timestamp': 1572299683.170468,
        'mac': '2c:0e:3d:3c:d2:64',
        'frequency': 2437,
        'sequence_number': 9488,
        'rssi': -71,
    },
    {
        'sensor_id': 4,
        'timestamp': 1572299679.186673,
        'mac': '2c:0e:3d:3c:d2:64',
        'frequency': 2437,
        'sequence_number': 9488,
        'rssi': -81,
    },
    {
        'sensor_id': 9,
        'timestamp': 1572299679.169782,
        'mac': 'ec:9b:f3:44:9b:6a',
        'frequency': 2437,
        'sequence_number': 65136,
        'rssi': -75,
    }
]
```

We also stored the RSSI data in an Amazon DynamoDB database [13] so we can later replay the data with different configurations. For some MAC addresses, we were able to trace back the emitting phones, but others were randomized. On average, a phone sends a probe request once every 5-6 seconds; however, the probe requests are mostly

inconsistent. To counter this, we specify a window size of 4-6 seconds according to the environment and the desired stride size. It is worth mentioning that even if a phone sends a probe request frequently, the signal may not necessarily be detected by every AP within range every time. Therefore, the smaller the window size, the more updated the localization is, but also, the lower the number of APs localizing it and the more prone it is to error. In the end, it is a trade-off.

3.3. Distance Estimation

Once we obtained the RSSI data from the MQTT topic, we estimated the distances between the phones and the APs using a fitted log-distance propagation curve. The log-distance propagation model is a derivative of the FSPL model [5], which we explained in Chapter 2. The log-distance model can be formulated as follows:

$$PL = PL_0 + 10\gamma \log_{10} \frac{d}{d_0}, \tag{3.1}$$

such that PL_0 is the path loss at a chosen reference distance d_0, and PL is the path loss at a distance d. We chose our reference distance d_0 to be 1 m. Solving for the distance d, the formula can be rewritten as follows:

$$d = 10^{\frac{PL - PL_0}{10\gamma}}. \tag{3.2}$$

PL can be substituted with the RSSI of the transmitter at the receiver. To find γ and PL_0, we must fit a curve. For that, we collected data from phones A, B, and C at varying distances (i.e., 0, 1, 2, ..., 12 m) at five different APs in the corridor, as demonstrated in Figure 3.3. We started at 0 m and collected data for roughly 8 minutes, noting down the start and the end timestamps before moving the phones a meter away from the APs and repeated. During this experiment, we collected a total of over 600 probe requests. That is approximately 50 probe requests at every distance.

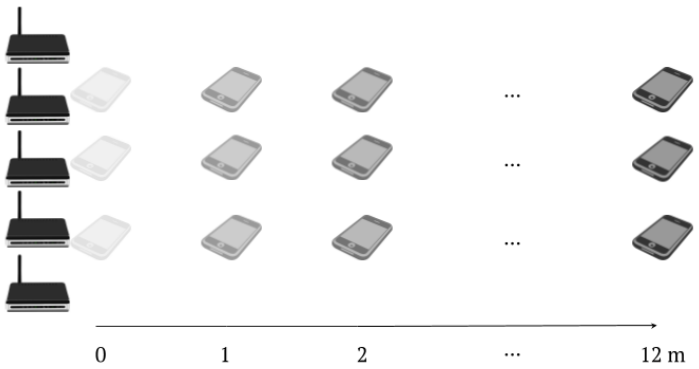


Figure (3.3) Five APs collecting RSSI data at various distances from three different phones to fit the path loss model.

Figure 3.4 reveals the raw data collected from 0 m up till 12 m. There is substantial noise in the data caused by power fluctuations as well as multi-path propagation. Nevertheless, a log-like curve structure can be distinguished as the RSSI variance significantly diminishes at distances $\geq$ 1 m. Studying the histogram in Figure 3.5, we can see this distinction even clearer between the RSSI values collected from a distance $=$ 0 m, and those collected from distances $\geq$ 1 m. Thus, we can understand why a log function is a good curve choice for a path-loss propagation model. These diagrams shed light on one of the biggest challenges in RSSI-based localization, which is the logarithmic decay of the signal. If we were to have a linear path-loss pattern, RSSI values would be less ambiguous, thereby making distance estimation task much easier. However, with the current propagation model, it is less likely to falsely estimate the distance of a phone with a strong RSSI of, for instance, -20 dBm than that of a phone with a weaker -60 dBm RSSI value.

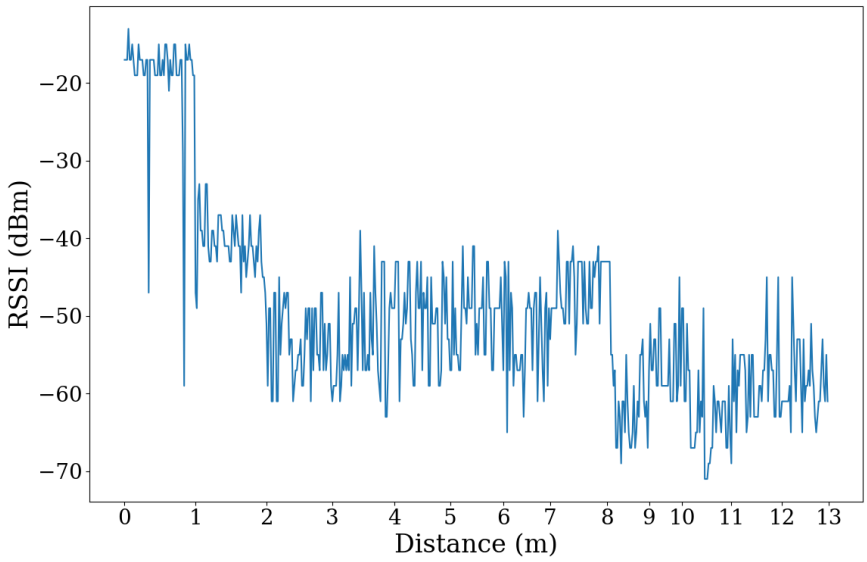


Figure (3.4) The raw data collected at varying distances to fit the log-distance curve.

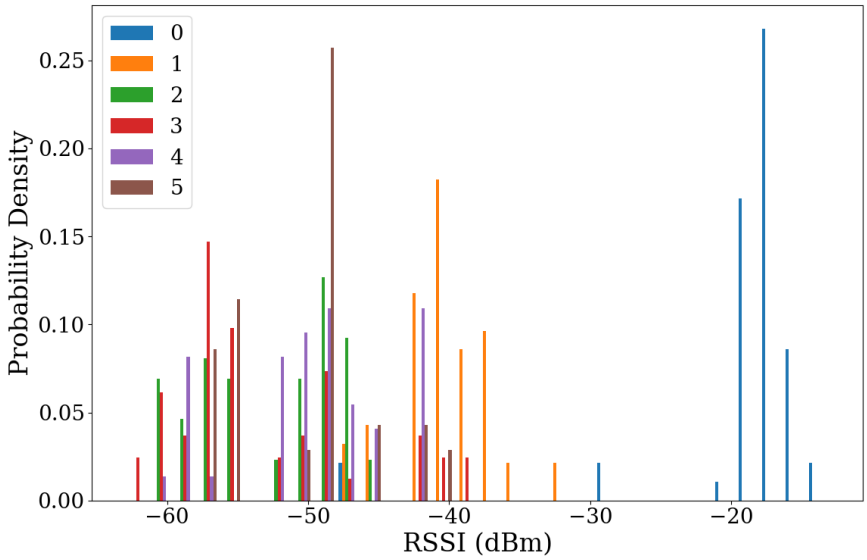


Figure (3.5) A distance color-coded histogram showing the RSSI probability density emitted by the phones at distances from 0 to 5 m.

Using the data collected from all three phones, we were able to fit the curve shown in Figure 3.6. The parameter values obtained from this are $PL_0 = -28.792$ and $\gamma = 3.191$. Using this curve, we can obtain a distance estimate of a given RSSI value.

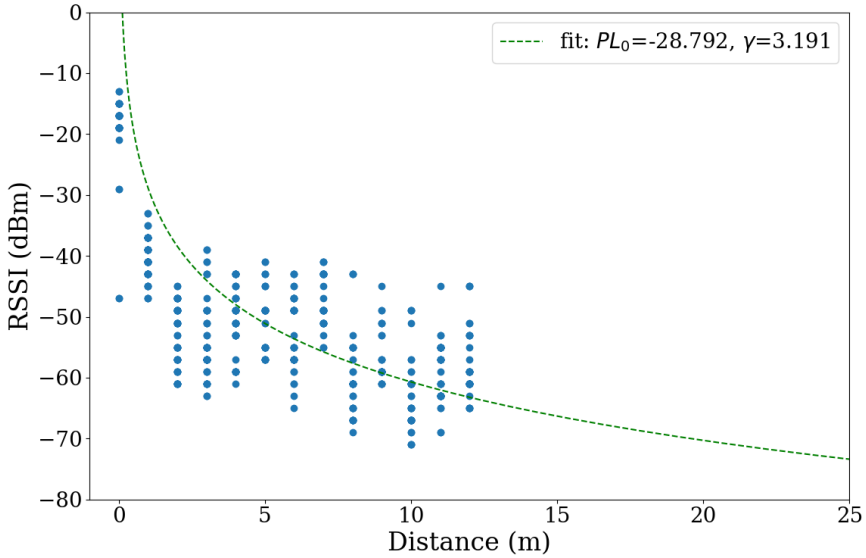


Figure (3.6) Log-distance curve fitted in our environment using data from three phones and five APs.

3.4. True Range Multilateration

3.4.1. Non-linear Least Squares

Now that we can estimate distances from individual APs, we can use those distances to do localization. Through trilateration, we can locate a phone with respect to exactly three APs of known locations. Since we are performing localization in a long corridor and eight rooms, using only three APs would not yield very good results. On the other hand, adding more APs leaves us with more equations than unknowns, which is what is known as an overdetermined system of equations:

$$\begin{aligned}
(x_p - x_1)^2 + (y_p - y_1)^2 &= {d_1}^2 \\
(x_p - x_2)^2 + (y_p - y_2)^2 &= {d_2}^2 \\
(x_p - x_3)^2 + (y_p - y_3)^2 &= {d_3}^2 \\
(x_p - x_4)^2 + (y_p - y_4)^2 &= {d_4}^2 \\
&\dots \\
(x_p - x_n)^2 + (y_p - y_n)^2 &= {d_n}^2.
\end{aligned} \tag{3.3}$$

For an overdetermined system, no solution exists unless in some specific cases which do not apply here. This is where multilateration comes in. True range multilateration aims to find the coordinates (x_p, y_p) of a desired object p, given the coordinates and the distances from n other objects, where $n \geq 3$. Unlike trilateration, multilateration relies on non-linear least squares (NLLS) to find a solution. NLLS is an iterative technique which tries to find the best possible solution given an overdetermined non-linear system of equations. It does so by approximating the problem to a linear problem and using an optimization algorithm of choice, iteratively minimizes the sum of squared residuals:

$$S = \sum_{i=1}^{n} [r_i - d_i]^2, \tag{3.4}$$

where r_i is the estimated distance and

$$d_i = \sqrt{(x_p - x_i)^2 + (y_p - y_i)^2}, \tag{3.5}$$

which is the Pythagorean distance between the object's location (x_p, y_p) and the AP's location (x_i, y_i) from which the measurement i was taken. The first step in the algorithm is to choose the initial guess, namely, the most likely location. That in itself is challenging because if we knew the location, we would not be running the algorithm in the first place. At first, we tried passing a random location as the initial guess. Later we came up with another idea, which provided a much more educated guess to the algorithm as an initial point to start: trilateration. The thing about the initial guess is that it needs not to be perfect; it just needs to be good enough. Thus, we selected the three APs with the closest estimated distances to trilaterate a preliminary estimate of the location before running the optimization.

3.4.2. Weighted Non-linear Least Squares

As we will later see in the evaluation, multilateration was an improvement over trilateration. However, there was still room for improvement. NLLS treated all APs equally, which given the logarithmic decay of signals indoors, was not the best approach. We have seen in Section 3.3 that distance estimation becomes more difficult the further the transmitter is from the receiver. Therefore, it makes more sense to give more weight to closer APs than ones that are further away; hence, the weighted non-linear least squares (WNLS).

In principle, the WNLS works in exactly the same way as its non-weighted counterpart. The only difference is we modify equation 3.4 such that each residual is multiplied by a weight w_i as follows:

$$S = \sum_{i=1}^{n} [w_i(r_i - d_i)]^2, \tag{3.6}$$

where

$$w_i = \frac{1}{{r_i}^2}, \tag{3.7}$$

which is the reciprocal of the square estimated distance. This gives closer APs more control over the localization outcome, which results in more accurate localization overall. We experimented with different weights, like the reciprocal estimated distance and the logarithmic estimated distance [14], but we found that the reciprocal of the square distance yields the best results.

3.5. Uncertainty Radius

To compute the uncertainty radius, we use the estimated distance from the nearest AP as well as information on the phone's localization history to compute the phone's speed. First, we set a threshold, τ_d for the minimum estimated distance and a speed threshold, τ_s, which is the maximum speed value in meters that we deem acceptable in our environment. The value of τ_d can be set with the help of the fitted log-distance curve (Figure 3.6). A good value, in this case, would be between 0 and 2 m, where the corresponding RSSI values are less ambiguous than those at larger distances.

The walking speed of the average person is between 1.21 m/s and 1.40 m/s [15], which varies according to the environment. For example, in a lecture hall, we expect the average speed of people to be close to zero, whereas, in a train station or an airport, the average speed is higher than average. Consequently, the value of τ_s should be set based on how dynamic the localization environment is.

Another parameter we need to set is the depth, δ. The value of δ is chosen based on how deep we want to go in the localization history. Typically, a value of 1 or 2 should

be enough, since older predictions are less relevant than recent ones, especially if the subject is in motion. Larger values may even be misleading since we do not account for the moving direction. Thus, if the person is moving back and forth around the same location, such information is not retained, making the true uncertainty more difficult to quantify.

Next, we define an empty array to store computed uncertainties. We then check our recorded localization history, $H_{1:t-1}$, which is a dictionary containing a series of localization predictions, indexed by the mac address of the phone, from phones detected in the past. Each prediction includes a location and a timestamp. If a history of the phone exists, we compute the distance d, and the time elapsed since the last prediction Δk and use the values to calculate the speed, s.

We then compute the uncertainty, u_t to be equal to the sum of the distance d from the last detection, and the maximum of the minimum estimated distance to the nearest AP and the distance threshold τ_d. This ensures a minimum uncertainty value, in case the distance to the AP is too small. We generally prefer the uncertainty to be overestimated rather than underestimated. After that, we check if the speed is within the defined speed threshold. If it is, we return the already computed u_t. Otherwise, there is a possibility that the prediction at $t-1$ was poor, and so we check for even older records. So, we store the computed prediction in the array before decrementing the value of the depth and removing the last element in $H_{1:t-1}$. We then repeat the previous steps until we find an older prediction that is within the speed threshold. Otherwise, the algorithm loops until either no more historical predictions are left, or until the maximum depth, δ is reached. We then assume that the last prediction was an anomaly, and we return the minimum computed uncertainty from the list.

This computation allows the radius to dynamically and robustly reflect the error. The value attained is used to visualize an uncertainty radius around the localization prediction on the map (Figure 3.7).

Algorithm 3 Uncertainty Radius Computation

Input: History $H_{1:t-1}$, MAC address mac, estimated distances D_t, location l_t, timestamp k_t, distance threshold τ_d, speed threshold τ_s, depth δ
Output: Uncertainty radius length u_t

function COMPUTE UNCERTAINTY($H_{1:t-1}, mac, D_t, l_t, k_t, \tau_d, \tau_s, \delta$)
  $uncertainties \leftarrow []$
  while $mac \in H_{1:t-1}$ & $\delta > 0$ **do**
    $l_{t-1} \leftarrow H_{1:t-1}[mac][t-1]$["location"]
    $d \leftarrow \text{distance}(l_t, l_{t-1})$ ▷ distance from historical prediction
    $k_{t-1} \leftarrow H_{1:t-1}[mac][t-1]$["timestamp"]
    $\Delta k \leftarrow k_t - k_{t-1}$ ▷ time since historical prediction
    $s \leftarrow d/\Delta k$ ▷ speed from historical to current location
    $u_t \leftarrow \max(\min(D_t), \tau_d) + d$
    if $s \leq \tau_s$ **then**
      return u_t
    end if
    $uncertainties \leftarrow uncertainties \cup [u_t]$
    $\delta \leftarrow \delta - 1$
    $H_{1:t-1} \leftarrow H_{1:t-2}$
  end while
  return $\min(uncertainties)$
end function

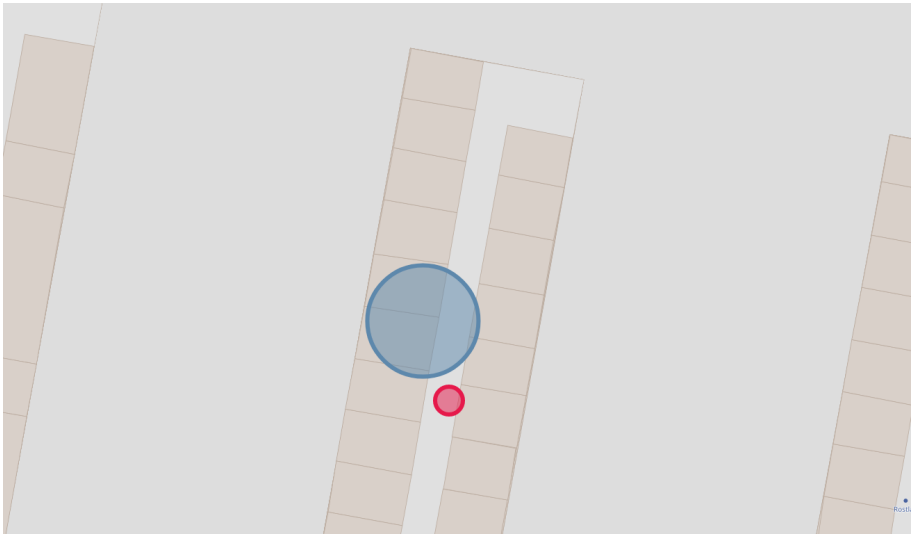


Figure (3.7) The localization predictions of two phones with uncertainty radii, visualized in OpenStreetMap. The prediction in red shows lower uncertainty than the one in blue.

3.6. Polygon Projection

One thing that can be useful is knowing the floor plan of the building where the indoor localization is taking place. Not only can it help us correct the locations of phones detected in inaccessible places, but it can also provide semantic knowledge about the rooms and areas in which people are spending their time.

To implement polygon projection, we first excluded any restricted rooms. We then segmented the rest of our localization area into nine polygons: eight rooms and one corridor. Since the rooms in the chair were already quite small, we did not need to segment every room further. For the segmentation, we used JOSM [16] (an OpenStreetMap editor) to trace the geographic outlines of the different rooms and obtain their physical coordinates. The polygons were exported as a list of JSON objects, which we used to create the polygons.

Listing (3.2) The format of a polygon JSON object.

```
{
    "type": "Room",
    "properties": {
        "level": "0",
        "ref": "00.11.065"
    },
    "coordinates": [
        [
```

```
            <vertix 1 latitude>,
            <vertix 1 longitude>
        ],
        [
            <vertix 2 latitude>,
            <vertix 2 longitude>
        ],
        [
            <vertix 3 latitude>,
            <vertix 3 longitude>
        ],
        [
            <vertix 4 latitude>,
            <vertix 4 longitude>
        ],
        [
            <vertix 1 latitude>,
            <vertix 1 longitude>
        ]
    }
}
```

The extracted polygons can be found in Figure 3.8. After computing a localization prediction, we checked if the predicted location was within the boundaries of any of the defined polygons. If it was, then the room in which the prediction was made was identified; otherwise, we projected the prediction to the nearest point on the nearest polygon. Most of the time, the predictions were already inside a polygon. However, every now and then, predictions were made in invalid locations (e.g., on walls or in restricted rooms). For those predictions, the location was projected to the closest accessible room. Other times, the localization was shifted completely outside of the building. In all cases, this procedure ensured that the localization prediction was inside a valid indoor location and returned the location's semantic label. Figure 3.9 summarizes the process in a flowchart.

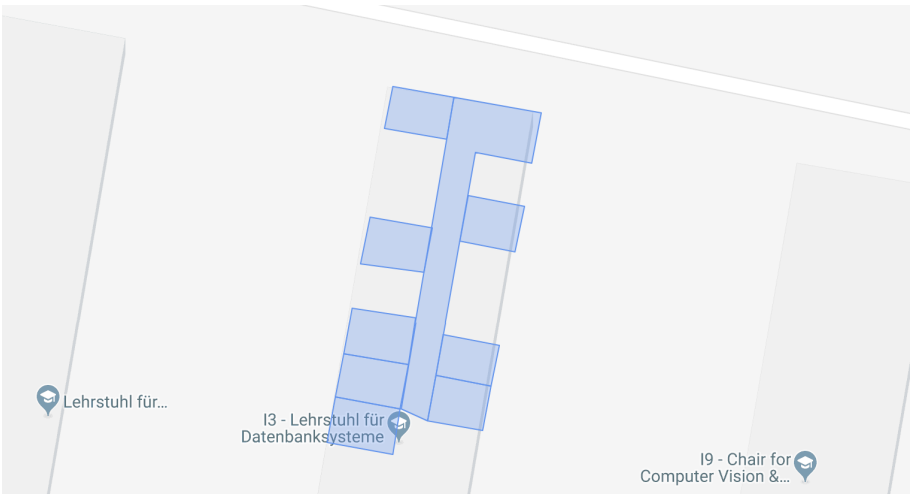


Figure (3.8) The localization area segmented into polygons based on topology, visualized in Google Maps. The gaps in the map are restricted area.

Through polygons, semantic information about the predictions was made available. Not only can we identify the room where a phone was detected, but we can even trace the path the phone has taken. However, we will see in Chapter 4 that localization error can limit the potential for improvement that polygon projection can offer. For instance, if a phone is being detected in the wrong but in a legal room, polygon projection can not help. Therefore, there needs to be a way to recognize localization error, from the semantic context of the prediction. Fortunately, there is a way to do so.

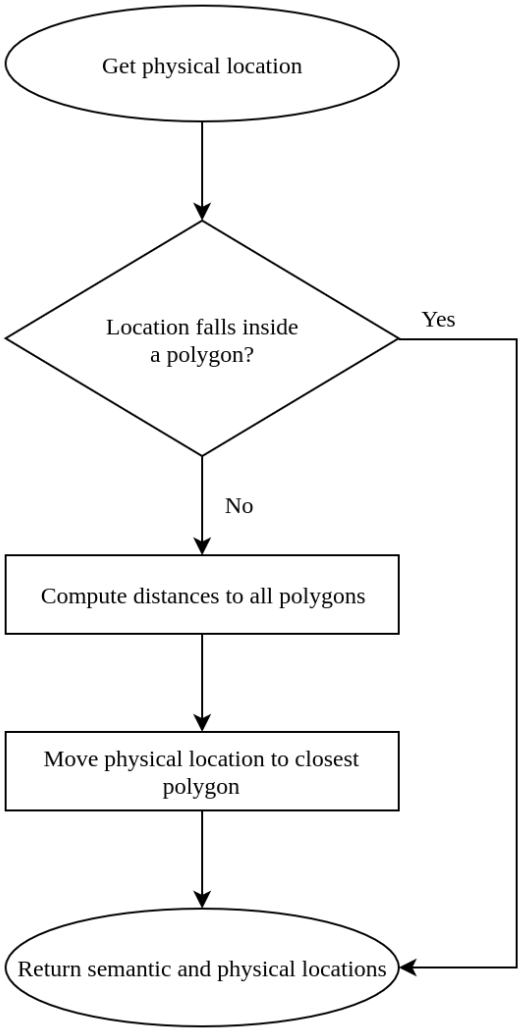


Figure (3.9) The polygon projection process flowchart.

3.7. Hidden Markov Model

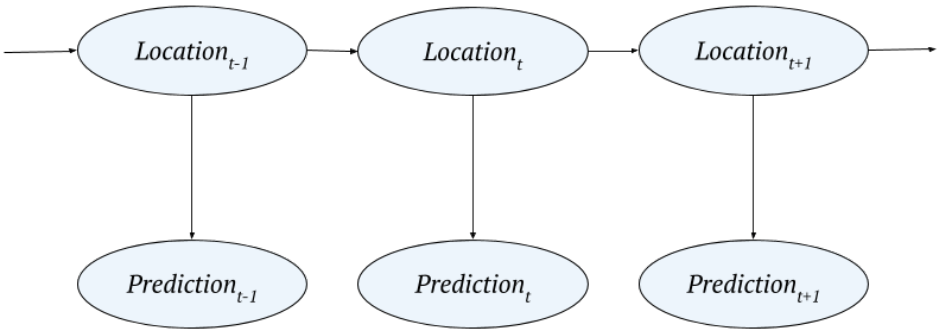


Figure (3.10) The HMM for localization which uses the real locations as the hidden states and the localization predictions as the observed states.

Having access to the semantic information of a localized point not only allows for a possibility of semantic analysis but also makes room for semantic correction. If a phone's localization predictions fluctuate between two rooms in a small time window, it is more likely that the person is in either of those rooms. Additionally, if a phone is frequently sending probe requests and is detected in two rooms, we would expect the person to be also intermediately detected in the corridor, especially if the rooms are not next to one another. This is the main motivation behind using HMMs. Using an HMM and our domain knowledge of the localization environment, we can model our physical space as an HMM where the rooms and the corridor are the hidden states, and the semantic localization predictions are the observed states. We can then use data collected in the area to fit an HMM, whose transition and emission probabilities reflect the real-life transition and localization patterns.

3.7.1. Initialization

First, we set the values for the start, transition, and emission probabilities between all the states based on our knowledge and after spending time in the localization area. The start probability distribution π was set as a uniform distribution since a phone was equally likely to start in any room. For the transition probability matrix A, we set the probability of staying in the same room to be 70%, going into the corridor to be 30% and 0% for going directly from one room to another. Given that some phones do not emit probe requests very frequently, this assumption is imperfect because a person may not be detected while in the corridor, going from one room to another. However, these are only

initial values which should be optimized after fitting the model to the collected data. The emission matrix B values were the most challenging to set because they needed careful examination of localization biases. For instance, persons in room 00.11.054 were more likely to be localized in room 00.11.056 than in room 00.11.054 because the rooms are right next to each other, and because unlike room 00.11.056, which was mostly empty, there were lots of objects (e.g., monitors and boxes) that blocked much of the signal detected by the APs in 00.11.054. Moreover, the rooms were designed such that the desks were taking up most of the central space, and personnel's chairs were located near the walls. We used this information to assign the prior emission probability distribution. The initial values for all three matrices were set as below. The labels starting with 'r' indicate a room (e.g., r59 corresponds to room 00.11.059), and 'c' is short for the corridor.

$$\pi = \begin{pmatrix} 0.111 & 0.111 & 0.111 & 0.111 & 0.111 & 0.111 & 0.111 & 0.111 & 0.111 \end{pmatrix}$$

$$A = \begin{array}{c} \\ r65 \\ r62 \\ r59 \\ r56 \\ r55 \\ r54 \\ r53 \\ r51 \\ c \end{array} \begin{array}{c} \begin{array}{ccccccccc} r65 & r62 & r59 & r56 & r55 & r54 & r53 & r51 & c \end{array} \\ \begin{pmatrix} 0.7 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3 \\ 0.0 & 0.7 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3 \\ 0.0 & 0.0 & 0.7 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.7 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.7 & 0.0 & 0.0 & 0.0 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.7 & 0.0 & 0.0 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.7 & 0.0 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.7 & 0.3 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.2 \end{pmatrix} \end{array}$$

$$B = \begin{array}{c} \\ r65 \\ r62 \\ r59 \\ r56 \\ r55 \\ r54 \\ r53 \\ r51 \\ c \end{array} \begin{array}{c} \begin{array}{ccccccccc} r65 & r62 & r59 & r56 & r55 & r54 & r53 & r51 & c \end{array} \\ \begin{pmatrix} 0.7 & 0.1 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.25 \\ 0.01 & 0.6 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.3 \\ 0.01 & 0.1 & 0.7 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.3 \\ 0.01 & 0.01 & 0.01 & 0.6 & 0.1 & 0.1 & 0.01 & 0.01 & 0.2 \\ 0.01 & 0.01 & 0.01 & 0.1 & 0.6 & 0.01 & 0.1 & 0.01 & 0.2 \\ 0.01 & 0.01 & 0.01 & 0.4 & 0.01 & 0.6 & 0.1 & 0.01 & 0.4 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.2 & 0.01 & 0.5 & 0.1 & 0.2 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.1 & 0.4 & 0.5 & 0.2 \\ 0.1 & 0.4 & 0.1 & 0.1 & 0.075 & 0.075 & 0.1 & 0.075 & 0.5 \end{pmatrix} \end{array}$$

3.7.2. Supervised Model

For supervised learning, we manually collected labeled data by walking through the localization space and recording the semantic truth labels, as well as the localization predictions after the polygon projection step. We managed to collect 500 labeled data samples by walking through the rooms and the corridor, using phones A and E. We then

split the data where 80% of the samples were used for training, and the remaining 20% were used for testing. We fit the model to our training data and decoded the test data using both the MAP and Viterbi decoders.

3.7.3. Unsupervised Model

For unsupervised learning, data labels are not needed. However, for the sake of performance evaluation, we used the same data set that we collected for the supervised model, and we trained an unsupervised model using the Baum-Welch algorithm. Same as the supervised model, we experimented with both the MAP and the Viterbi decoders to predict the test sequence. The results are discussed in the following chapter.

3.8. Fingerprinting

As previously explained in Chapter 2, fingerprinting is a method that uses probabilistic inference or machine learning to learn and identify the RSSI fingerprints of different physical locations. We implemented the latter. To do so, we gathered data samples in the localization space. Under the assumption that every room has its own fingerprint, we collected data separately in every room and the corridor. The features were normalized RSSI values that were detected by every AP, and the corresponding room labels were the predictions obtained through semantic localization. The data collection relatively easy since we collected data for one room at a time, and unlike the HMM data collection, the sequence did not matter. After collecting data in a room, we manually corrected any false semantic predictions, which were not so many. Therefore, we were able to collect 700 balanced and fully labeled RSSI fingerprints using phones A and C with little effort. Next, we trained several classifiers: a random forest (RF), a k-nearest neighbor (kNN), and a support vector machine (SVM) using different configurations. The kNN gave the best performance, so we discarded the other classifiers.

Since not all APs receive every emitted probe request, some of the RSSI features were missing. For those features, we used the last detected RSSI by the AP from the phone to be localized. In cases where there was no previous detection, we used a value of -60 dBm, until a signal was received. We chose this value because it had the highest probability in our collective RSSI distribution. It was crucial, however, to make sure that all APs were functional and streaming before we collected the data, otherwise, it can negatively impact the model's performance. A few samples from the data collected from 13 APs in one of the rooms is displayed in Listing 3.3.

Listing (3.3) Example of collected data samples in CSV format.

```
Room,AP1,AP2,AP3,AP4,AP5,AP6,AP7,AP8,AP9,AP10,AP11,AP12,AP13
00.11.053,-51,-63,-45,-45,-53,-55,-51,-75,-63,-31,-69,-45,-49
00.11.053,-53,-75,-45,-59,-63,-61,-49,-73,-69,-37,-69,-51,-59
00.11.053,-43,-69,-37,-43,-49,-61,-47,-69,-65,-37,-75,-47,-51
00.11.053,-63,-69,-61,-43,-49,-61,-59,-69,-65,-45,-75,-63,-61
00.11.053,-63,-69,-59,-49,-49,-61,-61,-77,-73,-67,-75,-53,-57
00.11.053,-57,-77,-65,-47,-65,-71,-59,-77,-73,-43,-75,-53,-57
00.11.053,-57,-77,-65,-47,-65,-71,-59,-77,-73,-43,-75,-53,-57
00.11.053,-49,-77,-55,-45,-59,-67,-55,-77,-73,-41,-73,-49,-63
00.11.053,-65,-81,-57,-57,-59,-75,-69,-77,-73,-57,-79,-59,-61
00.11.053,-47,-60,-51,-45,-47,-60,-45,-57,-53,-39,-73,-39,-43
```

3.9. Localization Publication

The final step in our approach is to publish the localization result through an MQTT topic. For this, we channel the data in the following JSON format:

Listing (3.4) Localization output as published on the MQTT topic.

```
message: {
    'macAddress': macAddress,
    'latitude': latitude,
    'longitude': longitude,
    'timestamp': timestamp,
    'uncertaintyRadius': uncertaintyRadius
}
```

The MQTT output is used by other systems that take care of anonymizing the data before it is visualized in a web interface. During experiments, the anonymization step was skipped, and the test phones' data was pushed to a Firebase NoSQL database instead. The anonymization and visualization processes are out of the scope of this thesis.

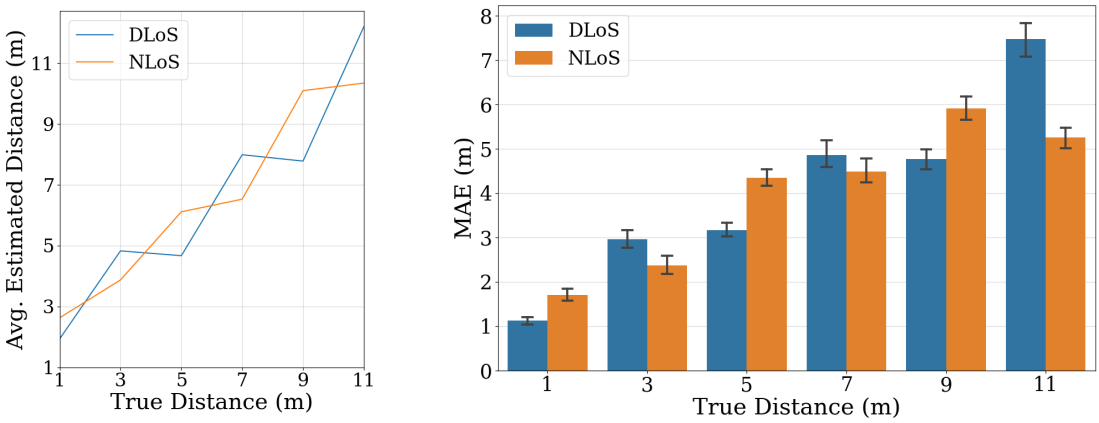
4. Evaluation and Discussion

In this chapter, we describe the experiments conducted to evaluate the performance of the different components of our system. All experiments were performed at Software & Systems Engineering chair of the Technical University of Munich campus using all or some of the phones we earlier presented in Table 3.1. The details of the experiments will be disclosed in detail, as well as the achieved results.

4.1. Distance Estimation

Estimating the distance is the first step towards a good location estimation. Therefore, we evaluate it independently to see how well it performs. To recall in Chapter 3, we fitted a log-distance model after collecting RSSI data from three different phones, at varying distances from five APs. Later we used this data to fit a curve, which we use for distance estimation of all detected phones from all deployed APs. To test the performance, we arranged a similar setup of the curve-fitting data collection where we aligned all five phones at distances 1, 3, 5, ..., 11 m from five different APs. Two of these phones were not used in the fitting of the curve, while all five APs were not. We then recorded the estimated distances from each phone to each of the five APs in DLoS and NLoS at every distance for 10 minutes. For NLoS, we did the same as DLoS, except that we also placed a metal sheet between the phones and the APs. The metal sheet was placed such that it was blocking the view between the phones and the APs.

In Figure 4.1, we can observe that the distance estimation in DLoS and NLoS is very comparable, and there is no significant difference, although we expected the estimation in NLoS to be consistently worse. The localization seems to even be better in NLoS setting in some cases. Thus, there did not seem to be a clear correlation between line-of-sight and distance estimation error in our environment. However, there was a positive correlation between the true distance and the estimation error, which comes as no surprise, given the log-distance attenuation of the signal over distance.



(a) (b)

Figure (4.1) In 4.1a, the mean estimated distance is plotted against the true distance in DLoS and NLoS. In 4.1b, the MAE at different distances in both settings is plotted.

4.2. True Range Multilateration

4.2.1. Baseline Experiment

To find a multilateration baseline, we collected RSSI data from all five phones in 10 different locations around the localization area. The 10 locations are marked in Figure 4.3. Knowing the exact locations' ground truths, we proceeded to run localization using different configurations. We then computed the mean and median distances between the true and the estimated locations for each of the 10 locations. The results are summarized in Table 4.1. The first thing to notice is that the mean error drops in half after switching from trilateration to multilateration. Contrary to our expectation, the trilateration initialization did not greatly improve the NLLS accuracy compared to using a random initialization. It even performed slightly worse in the case of WNLS. However, the WNLS consistently outperformed trilateration and NLLS, regardless of initialization. We use WNLS from now on as our baseline moving forward with the evaluation. The cumulative distribution function (CDF) of every phone localized using this baseline is plotted in Figure 4.2. The plot shows that most of the phones score an error of $\sim$ 3 m around the 80^{th} percentile and by the 90^{th} percentile, with the exception of Phone C, the phones are localized with an average error of $\sim$ 5 m.

The variance in error across locations can be explained by the AP coverage in each location (Figure 4.3). A more location-oriented evaluation can be found in Figure 4.4. The more marginal the location was, the higher the error. Closely studying the error distributions of the localization performance at locations 5 and 10, we can see the highest

Method	Initialization	Mean Error (m)	Median Error (m)
Trilateration	N/A	15.562	2.797
NLLS	Random	8.984	8.216
NLLS	Trilateration	8.690	7.778
WNLS	Random	2.088	1.428
WNLS	Trilateration	2.263	1.612

Table (4.1) *Physical localization performance*

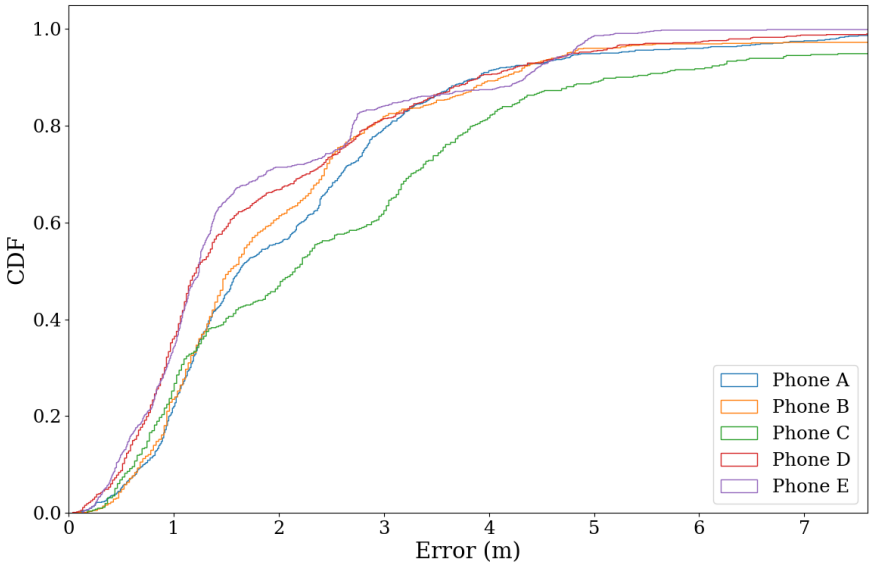


Figure (4.2) The CDF of the localization error obtained from using WNLS with random initialization in all 10 locations.

number of outliers. Looking at the map, one can easily see that there is one thing both locations have in common: they are both on either end of the corridor. Due to its length, the corridor is more liable to the multi-path problem than the rooms, which is a significant source of error. Moreover, the relatively poor coverage in these locations further contributes to a poor localization quality. Another thing to notice is the high median error at Location 3. The map shows Location 3 to be the only room with one AP, demonstrating that one AP in a room is not enough, regardless of the availability of more APs in nearby rooms and the corridor. In comparison, if we look at the rooms with three APs (e.g., locations 2 and 6), we can find a much better error distribution.

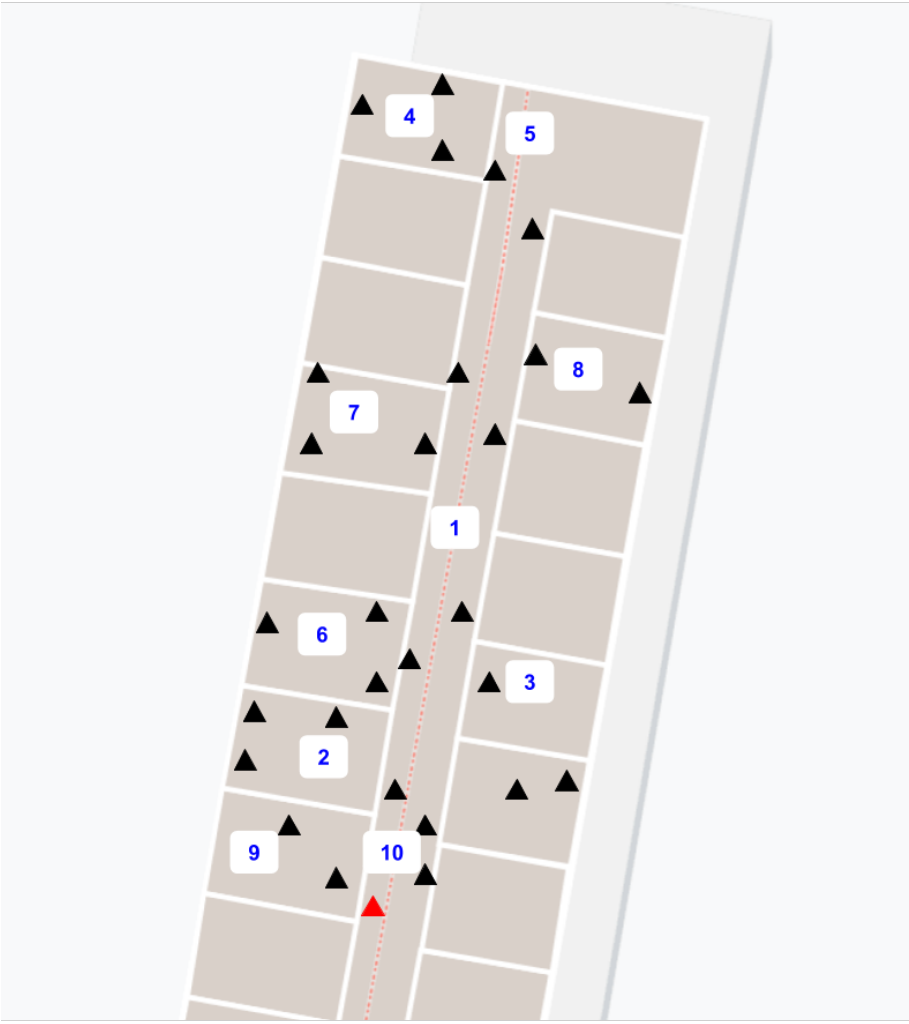


Figure (4.3) The 10 locations in the chair where data was collected to evaluate localization performance.

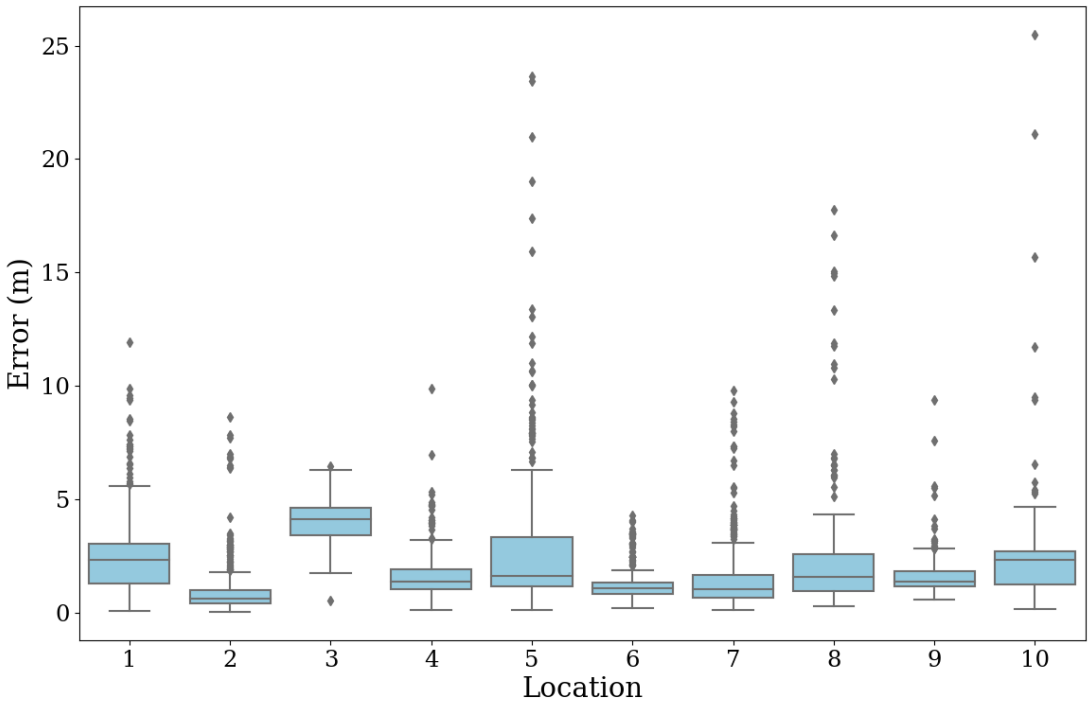


Figure (4.4) Boxplots showing the error distribution in the 10 different locations using WNLS with random initialization.

4.2.2. Point-of-Failure Experiment

Another interesting experiment that we conducted is what we call the point-of-failure experiment. The idea is to place a phone in the center of the localization area and perform localization n times, where n is equal to the number of APs subtracted by 3. With every iteration, we essentially turn off the closest AP until we are left with the three furthest APs. This experiment should help us understand how the WNLS localization quality deteriorates as the distance to the closest AP increases.

We performed this experiment using phones A, B, and E. We computed the centroid of the AP locations and placed the phones there to collect data continuously for 10 minutes. The centroid (x_c, y_c) is computed as follows:

$$(x_c, y_c) = \left(\frac{x_1 + x_2 + ... + x_k}{k}, \frac{y_1 + y_2 + ... + y_k}{k}\right), \tag{4.1}$$

where k is the number of APs. Then, we computed the true distances to all the APs and sorted them in ascending order. During the first iteration, we included all the APs in the localization. Gradually, we excluded the nearest APs, until we were left with only three, which is the minimum number of APs required to do localization in 2-D space as discussed in Chapter 2.

First, to eliminate any doubts concerning the role the number of connected APs plays in the localization performance, we ran the experiment once using only the closest three APs, and again using the furthest three, which should serve as a control. The closest three APs were at distances 2.40, 2.45, and 2.54 m, whereas the furthest three were at distances 13.28, 17.32, and 19.47 m from the phones. From the boxplots in Figure 4.5, one can witness a significant difference in the error distribution between both cases. Despite that only the three nearest APs were connected in the case on the left, the error remained considerably low with a median of 1.75 m compared to when the furthest three APs were connected, where the median error was 17 m. This demonstrates that the number of APs does not significantly matter when the APs are too far from the phone. Now that the aforementioned has been established, we can jump right into the experiment.

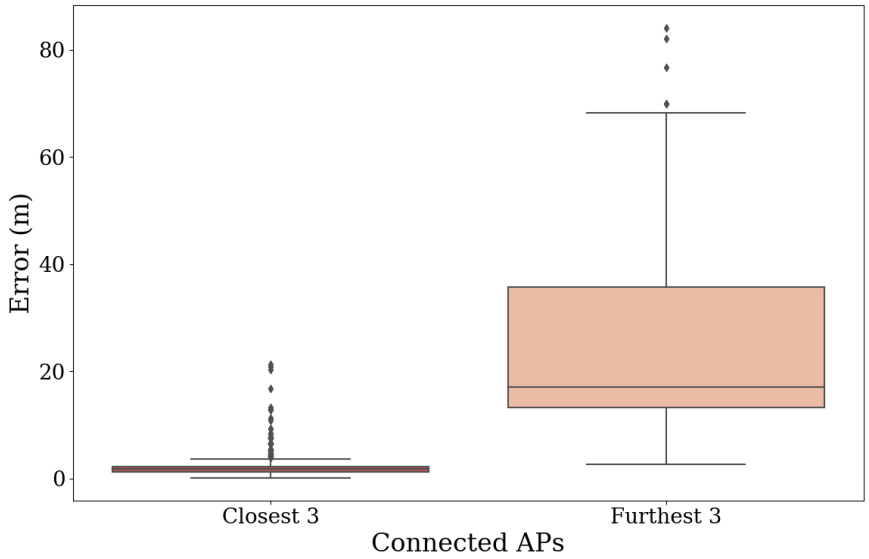


Figure (4.5) The localization error distribution in the cases when the closest three APs were connected, versus that in the case when the furthest three APs were connected.

We performed the main experiment as previously described, and Figure 4.6 illustrates how the error distribution is affected as we use less and less APs. It is nearly after $\sim$7 m do we start to see a rapid decline in the localization quality. Around 17 APs, the error spikes before it drops again for a brief number of rounds. We think this could be because of an imbalance in the distribution of APs surrounding the phones, where at 18 APs, two APs were tugging at the phones from opposite directions, virtually maintaining an equilibrium. However, after shutting down one of the APs, this equilibrium was lost, pulling the predictions in the direction of the second AP. However, this imbalance is quickly resolved after the second AP is also shut down.

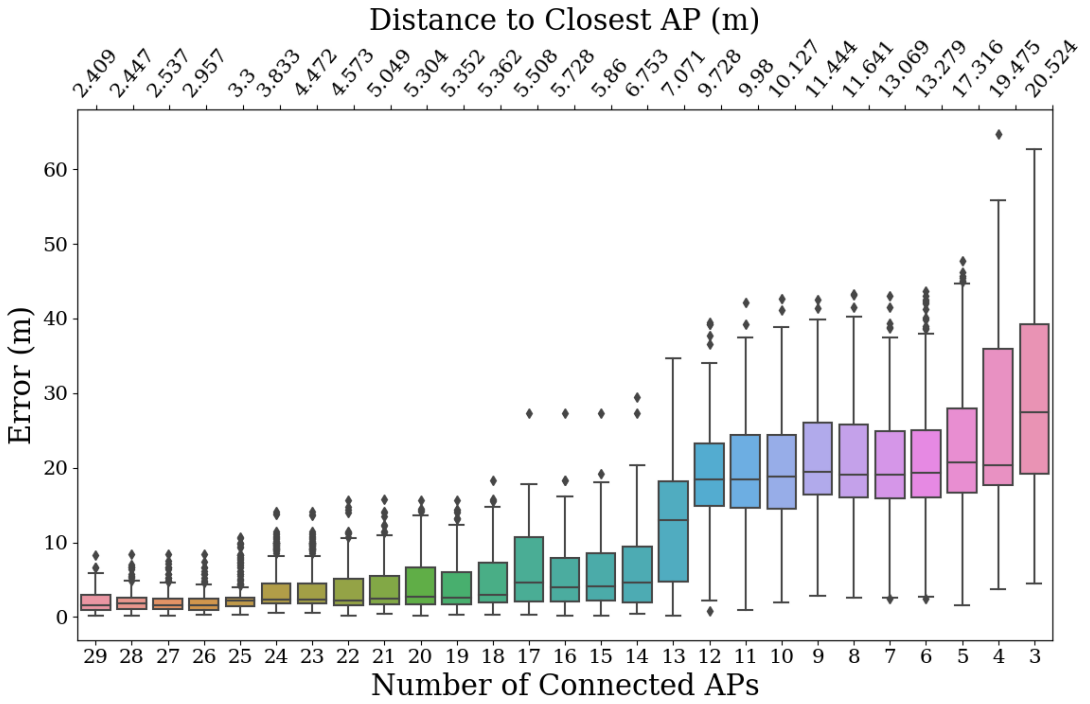


Figure (4.6) The error distribution plotted against the number of connected APs. The second x-axis contains the distance to the nearest AP in meters in every round.

4.2.3. RSSI Threshold Experiment

The weaker the RSSI, the harder the task becomes to estimate the real distance. We set out to investigate the effect of setting various thresholds on the RSSI values and studying their effect on the localization performance. We performed localization on the data samples we earlier collected from 10 different locations, using threshold values between -85 dBm and -45 dBm. Higher than -45 dBm, the localization predictions were so scarce, and so we did not go any further.

We found that the error consistently dropped in all 10 locations as the threshold increased, but at the cost of less frequent localization. Figure 4.7 shows how the localization frequency in five of the locations gradually decreases at first as the RSSI threshold is increased, until around -60 dBm, after which the localization frequency quickly drops.

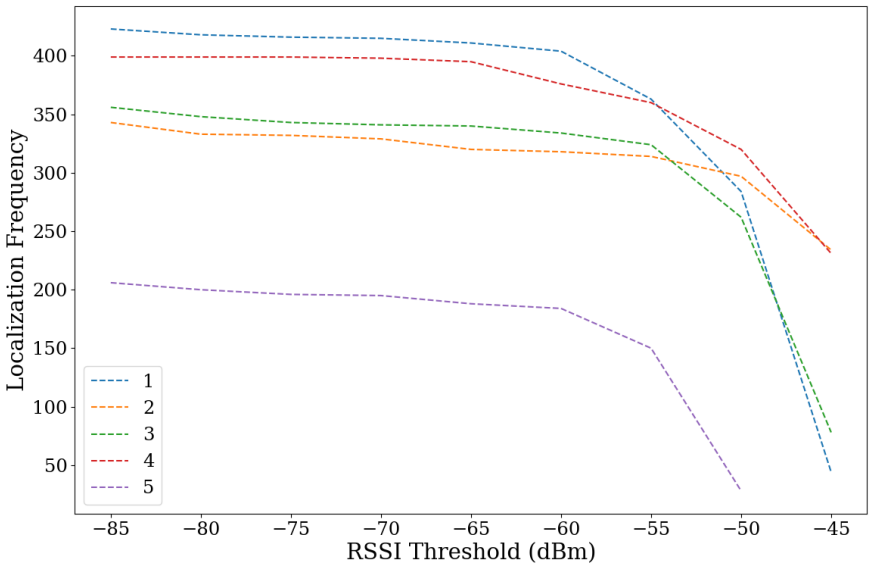


Figure (4.7) The localization frequency slowly declines as the RSSI threshold is increased from -85 to -60 dBm, before it rapidly drops around values $\geq$ -60 dBm in five different locations.

The CDF error of the different thresholds is plotted in Figure 4.8. The -45 dBm threshold is unmistakably better than the lower threshold values, resulting in less than 3 m accuracy at the 90^{th} percentile, compared to almost 5 m for the other thresholds. On the other hand, the error in lower thresholds does not seem to differ much. Therefore, if we do not care about having frequent localization predictions, -45 dBm is clearly a winner. However, such low prediction frequency is usually impractical. Thus, it can be seen that a threshold value between -60 dBm and -50 dBm generally provides a good cutoff without heavily compromising the localization frequency. Then again, it depends on the environment and the kind of trade-off we are willing to make.

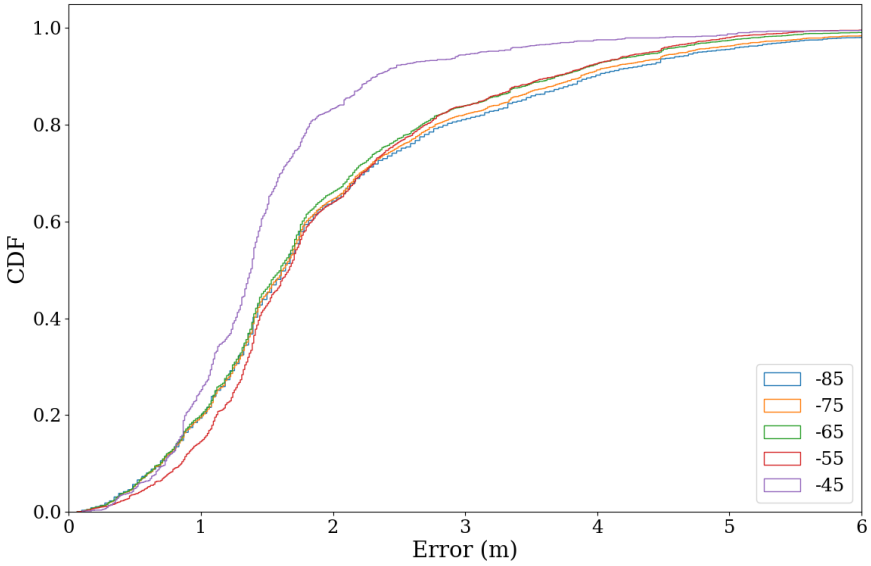


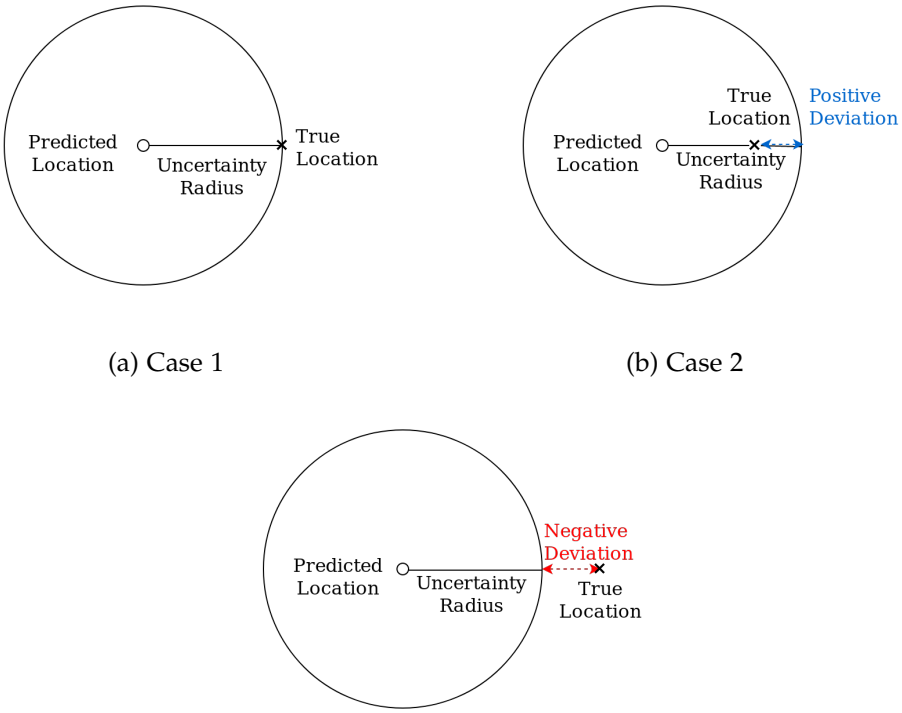
Figure (4.8) The error CDF produced from using different thresholds.

4.3. Uncertainty Radius

Besides multilateration, we evaluated how well the uncertainty radius was reflecting the actual error using phones all five phones. Ideally, the true location would lie on the uncertainty perimeter, but it is also acceptable if it lies within the uncertainty circle. We classify the possibilities into exactly three (Figure 4.9). In the first case, the true location lies exactly on the perimeter of the uncertainty circle. This was when the uncertainty computation algorithm was perfectly able to quantify the true error. In the second case, there is a positive deviation of the uncertainty perimeter from the true location. This means the error was overestimated. In the third case, the uncertainty radius falls short behind the true location. In this case, the uncertainty radius has underestimated the error.

To evaluate the uncertainty radius, we set the following parameter values: distance threshold $\tau_d = 1.5$, speed threshold $\tau_s = 0$, and depth $\delta = 2$. We then computed the MAE and the RMSE values for the deviation of the perimeter of the uncertainty circle from the true location of all five phones in the ten locations mentioned above. We were able to achieve an MAE value of **1.03 m** and an RMSE value of **1.79 m**. In the case of only positive deviation, we were able to achieve a mean deviation of **0.99 m**, and in that of negative deviation only, we achieved a mean deviation of up to **-1.08 m**. Figure 4.10 shows the CDF of the deviation of the uncertainty perimeter from the true location is an S-shaped curve centered around 0 m, which shows that the algorithm is almost just as likely to overestimate as it is to underestimate. This can be adjusted by increasing

the distance threshold τ_d by a value that is $\geq$ 1.08 m to minimize negative deviation. This would, of course, increase the positive deviation; however, as we mentioned before, overestimating the error is favored over underestimating it.



(a) Case 1

(b) Case 2

(c) Case 3

Figure (4.9) In case 1, the uncertainty radius is exactly equal to the localization error. In case 2, the error is overestimated resulting in positive deviation. In case 3, the error is underestimated resulting in negative deviation.

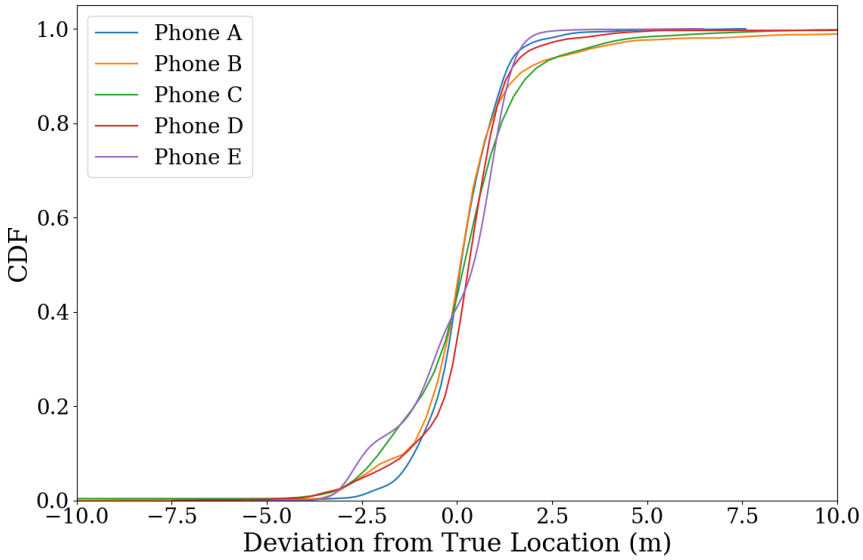


Figure (4.10) The CDF of the deviation of the uncertainty perimeter from the true locations of the phones in 10 locations.

4.4. Polygon Projection

To recall, polygon projection moves physical locations that fall in invalid regions to the closest defined polygon. Note that locations that already fall within a polygon are not affected by this. To measure the value that polygon projection adds to our system, we evaluated the physical and semantic localization error before and after the introduction of polygons using the data collected from the ten different locations. It is important to note that just because a point is closer to one polygon than another, does not mean that moving the point to that polygon is the best decision in every scenario. However, we found that in most cases, it does work and generally decreases the overall localization error, and increases the chances of a phone being detected in the correct room (Table 4.2). The results show that the polygon projection significantly reduces the physical error in the case of simple trilateration, to the point that it is even better than NLLS post the introduction of polygons. The semantic error, however, was not significantly improved (only 26.8% improvement). In both cases of NLLS, the physical error is also reduced, but not as much as trilateration. The semantic accuracy, however, was improved by approximately 36%. For WNLS with the random initialization, the physical error surprisingly slightly increased, while the semantic error was reduced by 56.9%. As for the WNLS with the trilateration initialization, the physical error was reduced to be even less than WNLS with the random initialization, and there was a 65% improvement in semantic localization.

Method	Initialization	No Polygons (m)		Polygons (m)		Correct Polygon
		Mean	Median	Mean	Median	
Trilateration	N/A	15.562	2.797	4.154	2.707	26.8%
NLLS	Random	8.984	8.216	5.307	4.454	36.3%
NLLS	Trilateration	8.690	7.778	5.071	4.286	36.8%
WNLS	Random	2.088	1.428	2.153	1.670	56.9%
WNLS	Trilateration	2.263	1.612	1.999	1.378	65.0%

Table (4.2) *Localization performance before and after polygon projection. The 'Correct Polygon' column contains the percentages of localization samples that were first localized in invalid locations, and were then projected onto the correct polygon.*

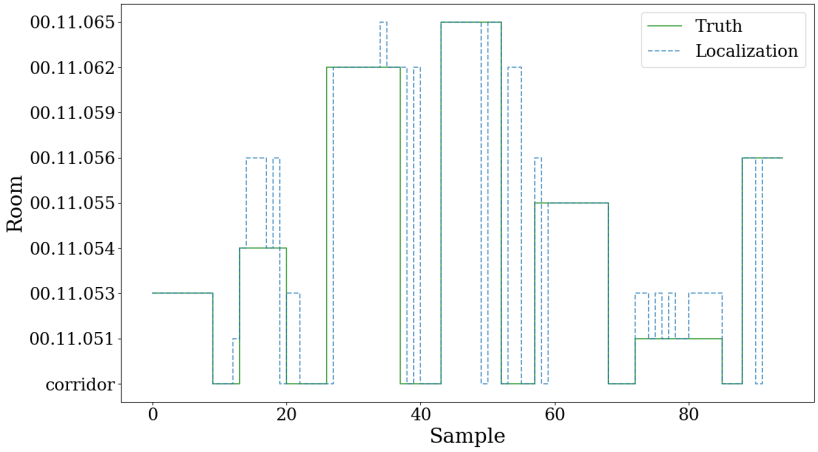
4.5. Hidden Markov Model

In Chapter 3, we explained how we attempted semantic correction by training an unsupervised HMM using the Baum-Welch algorithm, as well as a supervised HMM using labeled data collected using phones A and E. We tested both models on a test observation sequence composed of 100 samples using both the MAP and Viterbi decoders. We will focus on the supervised HMM with a MAP decoder, and the Baum-Welch trained HMM with a Viterbi decoder in our evaluation, since these two combinations of training/decoding algorithms performed the best. For readability, each of the following figures contains three sub-figures: the first sub-figure contains the truth sequence together with the localization sequence as predicted by the multilateration after polygon projection. The second sub-figure contains the HMM predictions together with the localization sequence. The third sub-figure contains the HMM predictions plotted with the ground truths. This format should help the reader understand and easily compare the performance of the different training and decoding pairs.

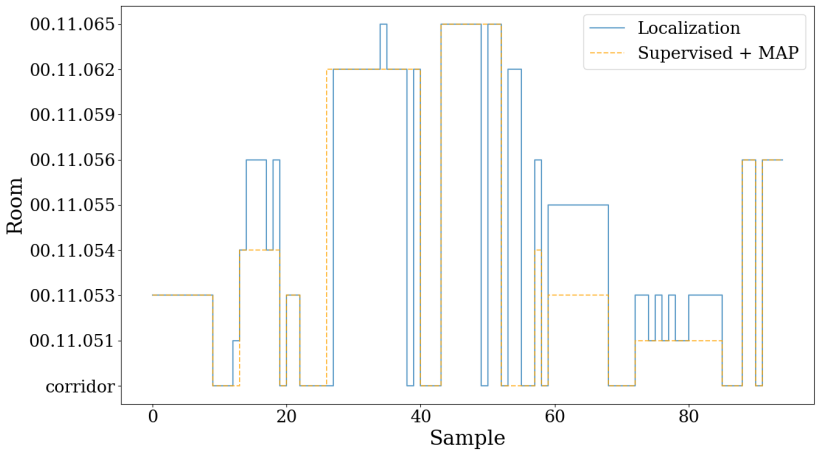
4.5.1. Supervised Model

In Figure 4.11a, the truth is plotted against the raw localization predictions. In hindsight, a notable amount of fluctuation can be observed in the localization predictions compared to the truth. Moreover, around samples number 20 and 80, clear biases can be seen as the localization fluctuates between exactly two rooms. As we previously mentioned in Chapter 3, some biases had emerged because of the topology of some rooms. In Figure 4.11b, the supervised HMM was introduced with the localization sequence decoded using MAP, plotted against the localization predictions. The first thing to notice is that the localization fluctuation is significantly diminished, thanks to the prior transition probability matrix that we defined earlier, which restricts any illegal transitions. The second thing is that in the case of the room biases, the HMM had favored Room 00.11.054

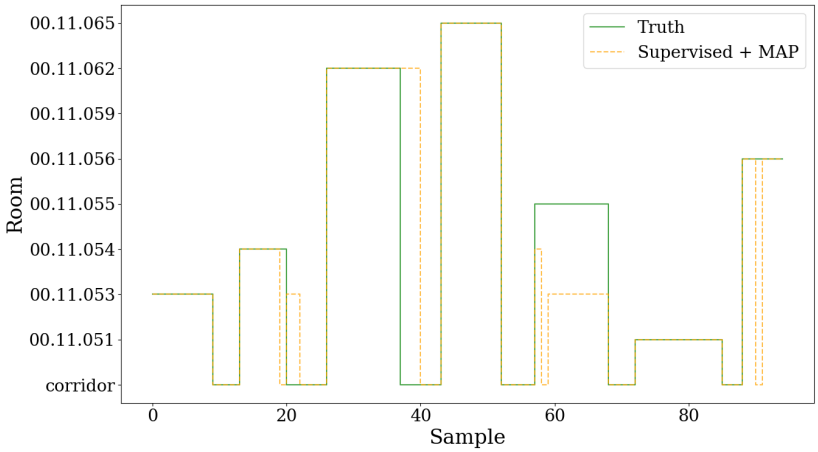
over Room 00.11.056 around Sample 20 and also favored Room 00.11.051 over Room 00.11.053 around Sample 80, even though in both cases, the frequency of localization predictions was higher in the latter. Now, if we look at Figure 4.11c, where the HMM predictions are plotted against the truth, we can see that the HMM has made the correct decision regarding those biases. This is due to the prior emission probability matrix that we defined, which provides the model with good enough initial probability estimates on a phone being falsely detected in every other room. This, together with the training data, help the model incorporate those biases in its predictions. The model, however, was not able to correct all the biases. For instance, the HMM was not able to correct the bias occurring around Sample 50 between rooms 00.11.053 and 00.11.055. The accuracy of the localization was improved from **0.716%** to **0.811%**. Despite the relatively small increase in accuracy, a lot of the fluctuation was removed, and most biases were corrected. Therefore, accuracy may not be the best metric to evaluate the performance of the HMM in this case.



(a) Truth vs. localization predictions



(b) Localization vs. HMM predictions



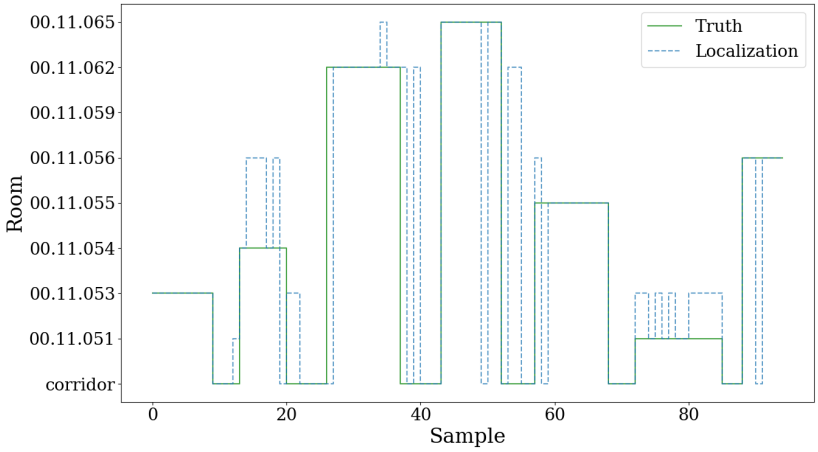
(c) Truth vs. HMM predictions

Figure (4.11) The time-series sequence of the semantic location of the surveyor as decoded by the supervised HMM using a MAP decoder.

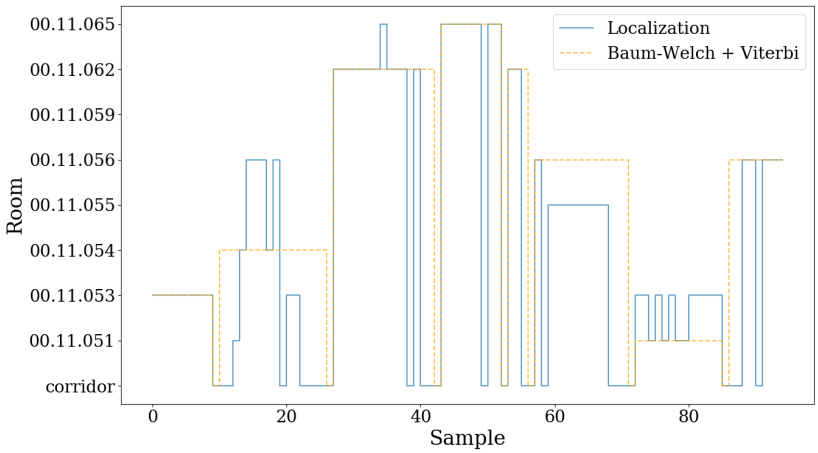
4.5.2. Unsupervised Model

Figure 4.12a again shows the ground truth plotted against the raw localization predictions. In Figure 4.12b, the unsupervised HMM was introduced, and the localization sequence was decoded using a Viterbi decoder. The predictions are plotted against the original localization predictions. We can see that the Viterbi algorithm similarly clears out the localization fluctuations. Note that the Viterbi decoder we are using uses MLE for parameter estimation, and so, unlike the MAP decoder, it does not utilize the prior probability distributions. Therefore, we had to make sure our training dataset contained data from all rooms. Otherwise, there is no way for the model to learn to predict a phone in an unseen room. Finally, Figure 4.12c shows the HMM predictions very close to the truth values with very brief transitions in the corridor. The model was able to learn this transition pattern well and predicted the phone in the corridor in-between rooms, despite those transitions being brief. The model, however, was not able to correct the biases towards rooms 00.11.059 and 00.11.056 around Sample 60. This is understandable since no prior emission distribution was used. Despite this, given that neither labels nor prior distributions were needed, the performance of the model was better than we expected.

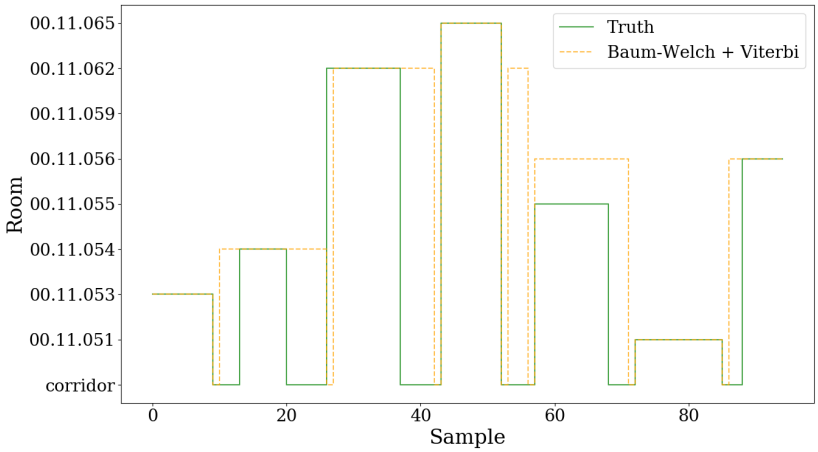
It is important to mention that the data collected and used in the experiment does not necessarily reflect reality, but mostly the surveyor's motion patterns, which were rather "staged" and may or may not mimic the real world. This is because a real person may spend hours in one room and one room only, except when leaving the room a few times through the corridor. A surveyor, however, must survey all rooms in a timely manner, and it is infeasible to collect labeled data in every room for hours, some of which are rarely occupied. Another thing to note is that phones do not emit probe requests at the same frequency, which can also be a source of error. The only way to have a scalable, robust HMM is to crowd-source large amounts of real unlabeled data over a long period of time from many different phones and use unsupervised learning to train the HMM. However, the performance of such a model without ground truths may be impossible to assess.



(a) Truth vs. localization predictions



(b) Localization vs. HMM predictions



(c) Truth vs. HMM predictions

Figure (4.12) The time-series sequence of the semantic location of the surveyor as decoded by the unsupervised HMM using a Viterbi decoder.

4.6. Fingerprinting

Finally, we evaluate the performance of our last component: fingerprinting. For this, we fit a kNN model using different values of k to our data and evaluated the prediction accuracy using 10-fold cross-validation. We found that $k = 3$ scored the best results (Figure 4.14). The model was able to identify the rooms and the corridor with a cross-validation score of over **91%**. The normalized confusion matrix in Figure 4.13 demonstrates the model's performance on every room. One can quickly notice the model's poor performance on the corridor compared to the rooms. This was quite expected since the corridor runs through the entire localization area and shares similar RSSI features with all the rooms. Such similarities vary based on where in the corridor a person is standing.

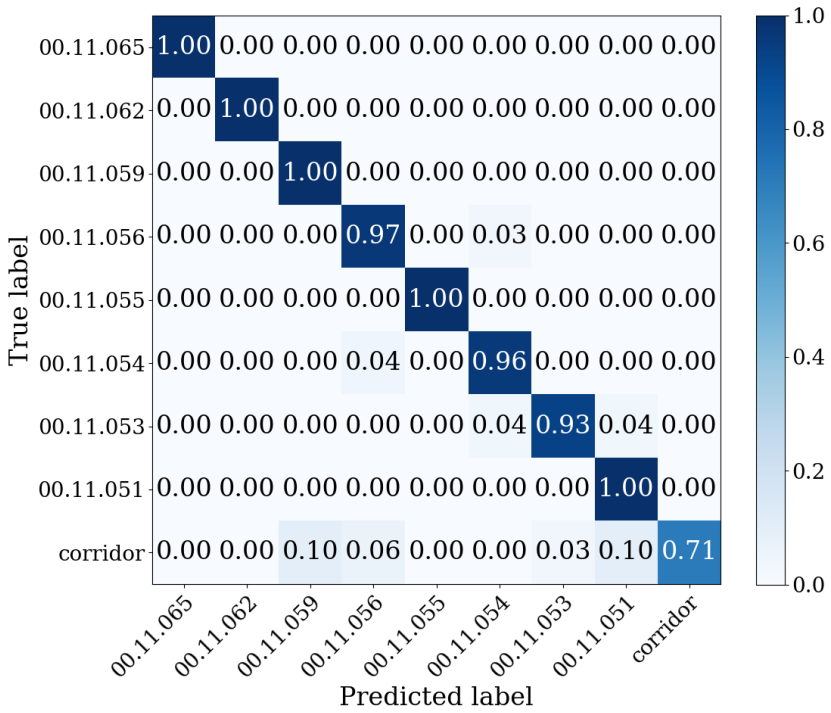


Figure (4.13) A normalized confusion matrix showing the kNN classification performance on rooms and corridor.

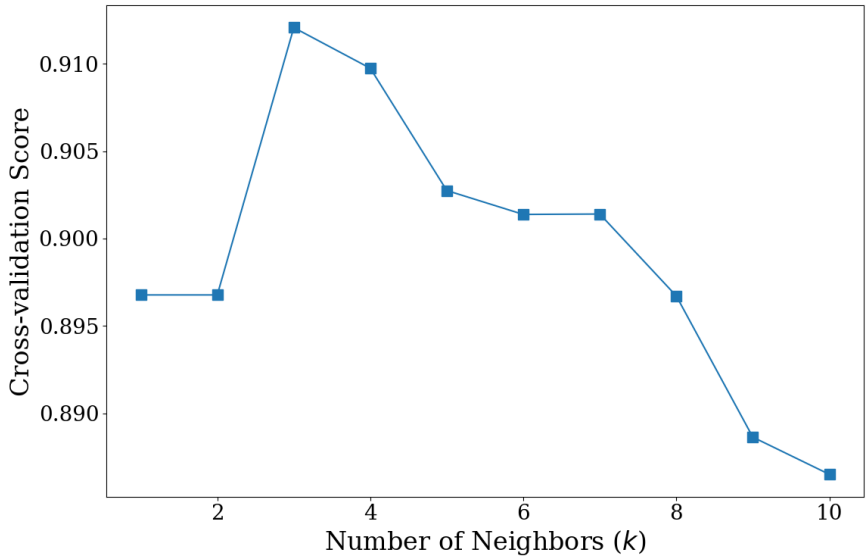


Figure (4.14) The cross-validation accuracy scored with different values of k.

Training a one-against-all kNN classifier and evaluating its performance on each room separately highlights this limitation even further. The ROC curve in Figure 4.15 shows the classifier scores the smallest area-under-curve (AUC) for the corridor compared to the rooms.

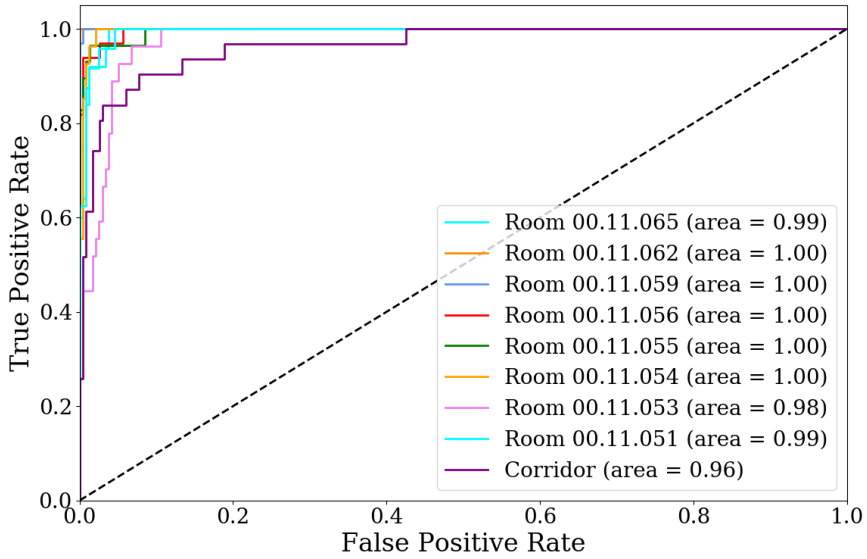
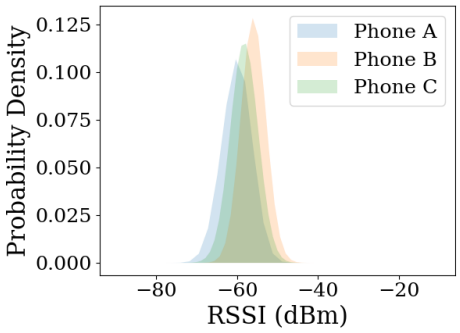


Figure (4.15) The ROC curves of the performance of a one-vs-all classifier on every room.

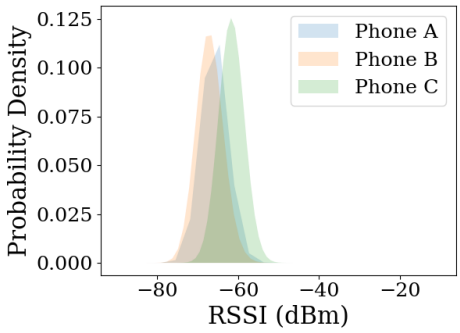
Despite the ease of the data collection process, this comes with a cost. Our fingerprinting approach does not predict the exact location of a phone inside a room, only the room. In order to obtain the physical locations, the IMU data of the phone used in the survey process needs to be collected along with the RSSI to estimate the number of steps the surveyor has walked from a known physical location. This requires the surveyor to walk at an even and concise pace, which is difficult and time-consuming. Another problem with fingerprinting is that the dataset can quickly become outdated if any change occurs in the environment or to the APs. This means that data needs to be recollected every time a broken AP is replaced or if a big object is moved, which is highly inconvenient. Our localization environment is relatively static; however, in highly dynamic environments, where there are many moving objects, fingerprinting may not be so effective. This raises the question of whether or not it is worth investing much time and effort in fingerprinting.

We also noticed that switching between phones while collecting data can confuse the model due to the so-called **device heterogeneity** problem. Essentially, every phone has a different wireless card with a different transmission power, which means that the model may end up learning every phone's fingerprint instead of the fingerprint of the room. This can cause behavior that is difficult to interpret and overall poor prediction performance. This incident led us to investigate the problem further, and so, we used the data collected from the experiment described in Figure 3.3, and plotted some RSSI kernel distributions for the three different phones at the five APs.

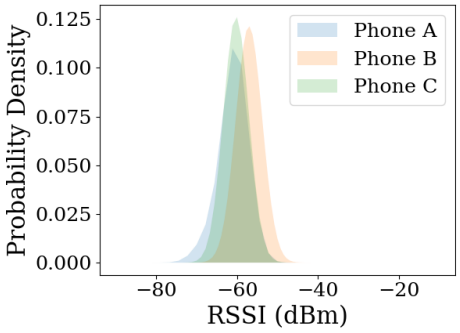
While examining the plots in Figure 4.16, it becomes clear that the RSSI distributions of the phones mostly overlap, with some slight shifts, especially in Figure 4.16e. Therefore, we can confirm that there is some diversity between the different phones. We could not, however, find a consistent shifting pattern between the distributions, which we can automatically correct. For instance, in Figure 4.16a, the RSSI distribution of Phone B seemed to shift towards smaller RSSI values with a peak close to -50 dBm, contrary to Figure 4.16b, where the distribution peaks close to -70 dBm. This is a relatively significant shift, which led us to further look into heterogeneity between the five different antennas that we used in the experiment. Plotting the distributions from the point-of-view of the APs shows that device heterogeneity is not only a transmitter phenomenon but is also a receiver phenomenon. Just like some phones have stronger transmission powers than other phones, different antennas do not detect the same signals from the same locations with the same strength. This is most prominent in Figure 4.17b.



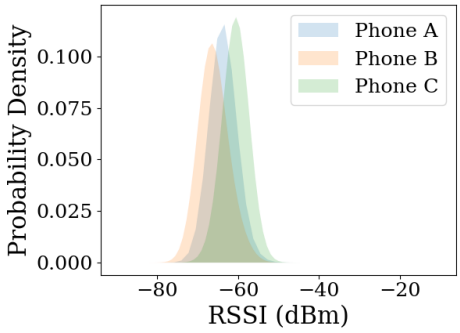
(a) RSSI distribution at AP 28



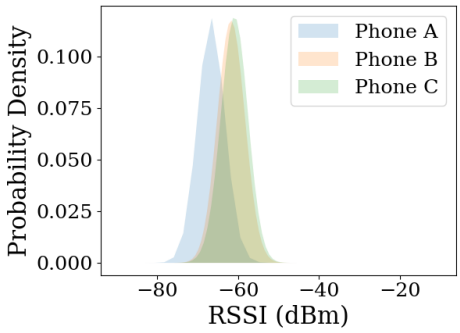
(b) RSSI distribution at AP 32



(c) RSSI distribution at AP 33

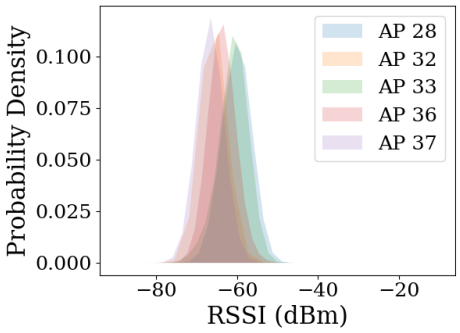


(d) RSSI distribution at AP 36

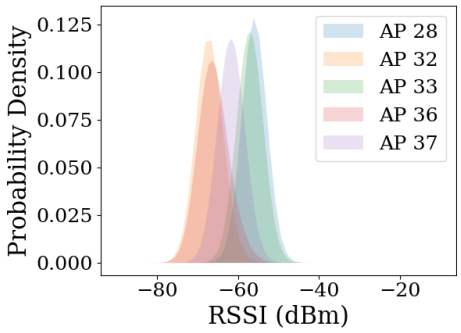


(e) RSSI distribution at AP 37

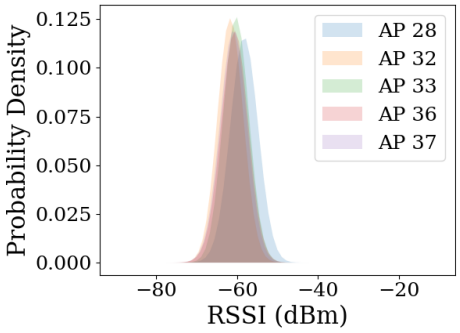
Figure (4.16) The RSSI Gaussian kernel distributions of phones A, B, and C as detected by five APs from the APs' perspective.



(a) RSSI distribution emitted by Phone A



(b) RSSI distribution emitted by Phone B



(c) RSSI distribution emitted by Phone C

Figure (4.17) The RSSI Gaussian kernel distributions of phones A, B, and C as detected by five APs from the phones' perspective.

In theory, there are several ways to overcome the device heterogeneity problem. One way is to calibrate the antennas using linear transformation, where a linear mapping function can be obtained through linear least-squares [17]. The RSSI distribution of an AP can then be shifted using the mapping function so that it is more aligned with the reference distribution. Due to the time constraint of this thesis, we preserve this problem for future work.

5. Related Work

In this chapter, we go through some of the related work in the area of indoor Wi-Fi localization, their advantages and disadvantages. Some relevant papers offer a brief overview of the various localization approaches [18], [19], [20], [21], [22], [23].

5.1. Geometry-based Techniques

We have already discussed trilateration as a basic geometry-based technique. There is also triangulation, which is the same as trilateration, but instead of distances, it uses angles. Microsoft's RADAR [24] is a prime example of triangulation using the log-distance estimation. In addition to triangulation, they implement fingerprinting using a *k*NN classifier to locate users with a 2-3 m median accuracy. They followed this up in a later paper with an enhancement to RADAR using a "Viterbi-like algorithm" [25] to mitigate localization fluctuations, and were able to improve the accuracy by over 33%. Anagnostopoulos et al. [26] alleviate the exponentially growing uncertainty of a propagating signal by utilizing the weighted average of the four closest APs to a phone to estimate its location. This technique ensures that the phone is localized inside the quadrilateral of the four nearest APs, allowing them to achieve up to 0.97 m accuracy for stationary phones and up to a 2 m accuracy for moving ones. Another geometry-based technique is the angle-of-arrival (AoA), which uses the angle between two nodes to estimate the distance between them. This method, however, is the least straight-forward of the formerly mentioned as it requires special hardware, such as array antennas.

5.2. Synchronization-based Techniques

An alternative approach is clock synchronization methods. Time difference of arrival (TDoA) requires that the APs are synchronized with one another before the arrival time delay between the APs is used to estimate the device's location. Other methods like time of arrival (ToA) require not only APs synchronization but also for the APs to be synchronized with the phone. Achieving perfect clock synchronization is somewhat challenging and often not done in Wi-Fi localization, as Wi-Fi bandwidths are not wide enough to obtain high accuracy [21]. ToA is rather more popular in ultra-wideband

(UWB), which like the name implies, has a wide bandwidth and is more suited for ToA. However, in both cases, ToA has special hardware demands and is also not feasible in passive localization, where accessing, much less configuring the phones is not an option. Nevertheless, ToA, when possible, has shown great results [27], [28].

5.3. Survey-based Techniques

Survey-based (or fingerprinting) approaches are another way to do localization, as we have seen in previous chapters. There are two ways to do fingerprinting: a probabilistic way and a machine-learning way.

5.3.1. Probabilistic Approach

One popular work is Horus [29], wherein the offline phase, RSSI samples were collected from the APs to construct a radio map corresponding to the localization space. During the online phase, they looked for the location L_i, which maximized the probability $P(L_i \mid s)$. Horus utilizes the correlation between RSSI samples from the same AP and uses multiple techniques to improve its performance. COMPASS [30] implements a probabilistic approach, which employs a phone's digital compass to detect the orientation of a person and account for the signal blocked, achieving a mean accuracy of 1.65 m. ARIEL [31] implements a room-based fingerprinting which uses clustering to identify Wi-Fi hotspots within rooms. It then applies Bayes' theorem to find the most likely room from a sequence of APs, sorted by RSSI values in descending order. Unlike our approach, ARIEL uses the IMU data of the phones to locate the fingerprints, but they do not require knowledge of the positions of the APs. They are able to achieve up to 95% room accuracy.

5.3.2. Machine Learning Approach

While the probabilistic method finds the location which scores the highest probability of an RSSI vector (or an APs vector in the case of ARIEL [31]), with the rising popularity of machine learning and deep learning, machine-learning-based fingerprinting is growing to become the standard survey-based approach. One example is Hallway [32], which constructs a floor plan from crowd-sourced IMU and RSSI data, and can identify hallways and room positions with 91% accuracy and their sizes with 66% accuracy. Tsiamitros et al. [33] implement a crowd-sourcing clustering technique that clusters regions with similar RSSI fingerprints together instead of being confined by a predefined floor-plan. They were able to achieve a mean error of 2.81 m.

5.4. Channel State Information

So far, we have discussed Wi-Fi localization work that uses RSSI values for location estimation. In the last few years, a new possibility has unfolded, making way for more fine-grained localization. The channel state information (CSI), which can be obtained from the physical layer, provides information on each sub-carrier in the frequency domain [34], [35]. Contrary to RSSI, it produces more than one value at a time, is more stable, and is less prone to the multi-path and NLoS problems. CSI can yield very high-resolution localization (up to a few centimeters accuracy), which is significantly better than RSSI. However, CSI can only be extracted through a handful of devices after some firmware modifications, primarily Intel's IWL 5300 [36], and Atheros 802.11n wireless chipsets [37]. Only recently have new tools started to emerge [38], allowing CSI to be extracted on more devices (e.g., Raspberry Pis). This is a big step that will revolutionize indoor localization and standardize fine-grained localization. Wang et al. implement a CSI-based fingerprinting solution (DeepFi) [39], where they train a deep neural network to learn to identify Wi-Fi fingerprints. They can obtain 90 CSI values from each packet, all of which are included in the fingerprinting and can localize with up to a 0.94 m accuracy. Other examples of work that adopted the CSI for fingerprinting are [40], [41], [42].

5.5. Device Heterogeneity

Some work investigates the effect of device heterogeneity in indoor localization. Park et al. [43] show that linear transformation of RSSI distributions alone is not enough, and demonstrate how kernel estimation can minimize noise. Combined with linear transformation, kernel estimation can produce better device calibration and reduce diversity.

5.6. Useful Applications

We investigated some work that uses passive localization in real-life. The first is Schmidt et al. [44], in which passive localization was used to detect and localize cellular signals in all of 2G, 3G and 4G bands inside a prison where mobile phones are of course prohibited. They implemented a fingerprinting approach that uses APs deployed on the outside of the building, where the APs could not be tampered with and were able to confiscate multiple phones. Another work is Montoliu et al. [45], where Wi-Fi fingerprinting was used to monitor older people's wellbeing in the comfort of their homes. The participants were asked to wear smartwatches that emitted Wi-Fi signals to be detected by deployed APs. The smartwatches helped track the participants' movement in different rooms and detect abnormal patterns.

6. Conclusion

Throughout this thesis, we presented our approach for implementing passive localization using Wi-Fi APs. In this chapter, we give a summary of our approach, and we discuss some possibilities for future work.

6.1. Summary

To summarize, we have discussed the problem of using GNSS for localization indoors and presented RF-based technologies as a more viable option. We specifically focused on Wi-Fi localization through RSSI log-distance estimation. We talked about the challenges of using RSSI and how we can best accommodate them by choosing a threshold. We also introduced multilateration using NLLS and discussed the limitation of NLLS in RSSI-based localization, which we alleviated by introducing a weight parameter to implement WNLS. We presented an uncertainty radius computation algorithm. Using map segmentation and polygon projection, we not only enhanced the physical localization performance, but we also acquired semantic knowledge about the predictions as well. Trained HMMs were used to correct the semantic fluctuations and biases in time-series prediction sequences. Finally, we described our room-based fingerprinting implementation and its limitations, as well as the device heterogeneity problem.

Furthermore, we evaluated the performance of our components through several experiments which we conducted at different locations in the chair. Moreover, we saw how areas with low AP coverage experienced poor localization quality. The point-of-failure experiment helped realize the minimum distance required from an AP to maintain a decent localization quality. We also discussed the consequences of setting high RSSI thresholds on the localization quality, including the unwanted side-effect of reduced prediction frequency.

In conclusion, passive localization can be used to give governments and businesses insights into how to better design cities and infrastructures to improve their quality and efficiency, thus improving our quality of life. By monitoring human traffic inside shopping malls, train stations, and airports, organizations can learn how to optimize their infrastructures to cut down on costs and improve the quality of their services. Passive indoor localization can also be used to detect wireless devices in device-restricted

areas. It goes without saying that regulating such technology using laws and legislations should be a priority, to ensure the anonymity of the data, and the privacy and safety of the public.

6.2. Future Work

There yet remain some open questions that we would like to answer in the near future. For one, is there a more efficient way to collect labeled data for the supervised HMM? Additionally, how well would our fingerprinting model generalize for devices that were not used in our experiments? For the model's inadequate performance on classifying the corridor, we would like to try segmenting the corridor into smaller polygons, such that each section can have a more distinctive RSSI fingerprint, retrain the model and reevaluate the performance to see if this limitation is mitigated. The uncertainty radius computation can most certainly be improved. One idea is to consider more APs when computing the uncertainty radius. We also intend to test our system with more than just five phones from more phone manufacturers, with different types of AP antennas and in more dynamic spaces. As for the device heterogeneity problem, we wish to find a solution to this problem with minimal calibration effort. Extending the system to include the 5 GHz frequency band as well as 3-D localization in multiple-floor buildings is something we look forward to doing. Other than that, we believe there is higher potential in CSI over RSSI, and we would prefer switching to CSI at some point in the near future.

Appendix A: Implementation Notes

In Chapter 3, we gave a detailed description of our approach. Here, we dive into technical details regarding the decisions made concerning our choice of libraries. Our localization system was entirely implemented in Python 3.6. The reason we chose Python is that our approach involves several mathematical and machine learning algorithms that are well supported by a range of popular Python libraries, such as NumPy, SciPy, Scikit-learn, Matlibplot, and Seaborn. A great alternative to Python would be R.

A.1. Data Handling

To configure MQTT, we used the AWSIoTPythonSDK [12] library. For running experiments, we stored and queried the RSSI data from the DynamoDB using the Boto 3 package [46]. The library Pyrebase [47] was used to connect and write the localization data to Firebase for real-time visualization. We also used Pandas [48] to construct DataFrames from the historical data as needed.

A.2. Physical Localization

To fit the log-distance curve, we used the curve_fit() function from the library SciPy's optimize package. The WNLS was implemented using the least_squares() package, also from SciPy. The optimization method used for WNLS was the default trust region reflective algorithm (TRF).

A.3. Polygon Projection

We used the library Shapely [49] to implement the polygon projection, as it provides the possibility to create Point and Polygon objects. Moreover, Shapely has a distance() function, which computes the closest distance from two shapes (e.g., a point and a polygon), which was valuable for our polygon projection implementation. Gmaps [50] was used to visualize the polygons in Google Maps, ensuring the outlines of the polygons

aligned with the rooms as intended.

A.4. Hidden Markov Model

For the HMMs, we tried several libraries like HMMlearn, Seqlearn, and Pomegranate [51], from which we decided to use Pomegranate for its support for both supervised and unsupervised learning. Pomegranate enabled us to initialize a HMM using the HiddenMarkovModel.from_matrix() Function, which takes as parameters a list of states, as well as start, transition, and emission matrices. Using the function fit (), which takes the training data and the type of model we want to train (i.e., Baum-Welch, labeled or Viterbi), we were able to train both the supervised and the unsupervised models. The library also provides a predict() function, which expects a sequence of observations to be decoded, as well as the desired decoding algorithm (i.e., Viterbi or MAP).

A.5. Fingerprinting

To implement fingerprinting, we heavily relied on Scikit-learn since they offer a wide variety of relevant packages, such as the KNearestNeighbor package, which we used to train the *k*NN classifier, as well as the preprocessing package for feature normalization. Moreover, we used Scikit-learn's auc and roc_curve packages to plot the ROC curves. The package cross_val_score was also used to evaluate the cross-validation accuracy score of the model.

Bibliography

[1] Neil E Klepeis et al. "The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants." In: *Journal of Exposure Science and Environmental Epidemiology* 11.3 (2001), p. 231.

[2] Dave Evans. "The internet of things: How the next evolution of the internet is changing everything." In: *CISCO white paper* 1.2011 (2011), pp. 1–11.

[3] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins. *Global positioning system: theory and practice*. Springer Science & Business Media, 2012.

[4] Kamol Kaemarungsi. "Distribution of WLAN received signal strength indication for indoor location determination." In: *2006 1st International Symposium on Wireless Pervasive Computing*. IEEE. 2006, 6–pp.

[5] Harald T Friis. "A note on a simple transmission formula." In: *Proceedings of the IRE* 34.5 (1946), pp. 254–256.

[6] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[7] Suining He and S-H Gary Chan. "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons." In: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 466–490.

[8] Georgios Pipelidis and Christian Prehofer. "Models and tools for indoor maps." In: *Digital Mobility Platforms and Ecosystems* (2016), p. 154.

[9] Nadav Levanon. "Lowest GDOP in 2-D scenarios." In: *IEE Proceedings-radar, sonar and navigation* 147.3 (2000), pp. 149–155.

[10] *OpenStreetMap*. `https://www.openstreetmap.org/`.

[11] *MQTT*. `http://mqtt.org/`.

[12] *AWS IoT*. `https://aws.amazon.com/iot/`.

[13] *Amazon DynamoDB*. `https://aws.amazon.com/dynamodb/`.

[14] William Navidi, William S Murphy Jr, and Willy Hereman. "Statistical methods in surveying by trilateration." In: *Computational statistics & data analysis* 27.2 (1998), pp. 209–227.

[15] Betty J Mohler et al. "Visual flow influences gait transition speed and preferred walking speed." In: *Experimental brain research* 181.2 (2007), pp. 221–228.

[16] *JOSM*. https://josm.openstreetmap.de/.

[17] Andreas Haeberlen et al. "Practical robust localization over large-scale 802.11 wireless networks." In: *Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM. 2004, pp. 70–84.

[18] Hui Liu et al. "Survey of wireless indoor positioning techniques and systems." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.6 (2007), pp. 1067–1080.

[19] Yanying Gu, Anthony Lo, and Ignas Niemegeers. "A survey of indoor positioning systems for wireless personal networks." In: *IEEE Communications surveys & tutorials* 11.1 (2009), pp. 13–32.

[20] Da Zhang et al. "Localization technologies for indoor human tracking." In: *2010 5th International Conference on Future Information Technology*. IEEE. 2010, pp. 1–6.

[21] Chouchang Yang and Huai-Rong Shao. "WiFi-based indoor positioning." In: *IEEE Communications Magazine* 53.3 (2015), pp. 150–157.

[22] Zahid Farid, Rosdiadee Nordin, and Mahamod Ismail. "Recent advances in wireless indoor localization techniques and system." In: *Journal of Computer Networks and Communications* 2013 (2013).

[23] Ali Yassin et al. "Recent advances in indoor localization: A survey on theoretical approaches and applications." In: *IEEE Communications Surveys & Tutorials* 19.2 (2016), pp. 1327–1346.

[24] Paramvir Bahl et al. "RADAR: An in-building RF-based user location and tracking system." In: (2000).

[25] Victor Bahl and Venkat Padmanabhan. "Enhancements to the RADAR user location and tracking system." In: (2000).

[26] Grigorios G Anagnostopoulos and Michel Deriaz. "Accuracy enhancements in indoor localization with the weighted average technique." In: *SENSORCOMM* 2014 (2014), pp. 112–116.

[27] David Humphrey and Mark Hedley. "Super-resolution time of arrival for indoor localization." In: *2008 IEEE International Conference on Communications*. IEEE. 2008, pp. 3286–3290.

[28] Mark Hedley et al. "A platform for radio location research in ad hoc and sensor networks." In: *2007 International Symposium on Communications and Information Technologies*. IEEE. 2007, pp. 876–881.

[29] Moustafa Youssef and Ashok Agrawala. "The Horus WLAN location determination system." In: *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM. 2005, pp. 205–218.

[30] Thomas King et al. "COMPASS: A probabilistic indoor positioning system based on 802.11 and digital compasses." In: *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*. ACM. 2006, pp. 34–40.

[31] Yifei Jiang et al. "Ariel: Automatic wi-fi based room fingerprinting for indoor localization." In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM. 2012, pp. 441–450.

[32] Yifei Jiang et al. "Hallway based automatic indoor floorplan construction using room fingerprints." In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM. 2013, pp. 315–324.

[33] Nikolaos Tsiamitros et al. "Cross-Device Radio Map Generation via Crowdsourcing." In: *International conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2019.

[34] Kaishun Wu et al. "CSI-based indoor localization." In: *IEEE Transactions on Parallel and Distributed Systems* 24.7 (2012), pp. 1300–1309.

[35] Zheng Yang, Zimu Zhou, and Yunhao Liu. "From RSSI to CSI: Indoor localization via channel response." In: *ACM Computing Surveys (CSUR)* 46.2 (2013), p. 25.

[36] Daniel Halperin et al. "Tool release: Gathering 802.11 n traces with channel state information." In: *ACM SIGCOMM Computer Communication Review* 41.1 (2011), pp. 53–53.

[37] Yaxiong Xie, Zhenjiang Li, and Mo Li. "Precise Power Delay Profiling with Commodity WiFi." In: *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. MobiCom '15. Paris, France: ACM, 2015, pp. 53–64. ISBN: 978-1-4503-3619-2. DOI: 10.1145/2789168.2790124. URL: http://doi.acm.org/10.1145/2789168.2790124.

[38] Francesco Gringoli et al. "Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets." In: *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*. ACM. 2019, pp. 21–28.

[39] Xuyu Wang et al. "CSI-based fingerprinting for indoor localization: A deep learning approach." In: *IEEE Transactions on Vehicular Technology* 66.1 (2016), pp. 763–776.

[40] Jiang Xiao et al. "FIFS: Fine-grained indoor fingerprinting system." In: *2012 21st international conference on computer communications and networks (ICCCN)*. IEEE. 2012, pp. 1–7.

[41] Kaishun Wu et al. "Fila: Fine-grained indoor localization." In: *2012 Proceedings IEEE INFOCOM*. IEEE. 2012, pp. 2210–2218.

[42] Xuyu Wang, Lingjun Gao, and Shiwen Mao. "CSI phase fingerprinting for indoor localization with a deep learning approach." In: *IEEE Internet of Things Journal* 3.6 (2016), pp. 1113–1123.

[43] Jun-geun Park et al. "Implications of device diversity for organic localization." In: *2011 Proceedings IEEE INFOCOM*. IEEE. 2011, pp. 3182–3190.

[44] Armin Schmidt et al. "Indoor Tracking and Localization of non-authorized Cellphones in Prisons Using Uplink Signals." In: *International conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2019.

[45] Raul Montoliu et al. "Senior Monitoring: A Real Case of Applying a WiFi Fingerprinting-based Indoor Positioning Method for People Monitoring." In: *International conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2019.

[46] *Boto 3*. `https://boto3.amazonaws.com/v1/documentation/api/latest/index.html`.

[47] *Pyrebase*. `https://pypi.org/project/Pyrebase/`.

[48] *Pandas*. `https://pandas.pydata.org/`.

[49] *Shapely*. `https://pypi.org/project/Shapely/`.

[50] *Gmaps*. `https://pypi.org/project/gmaps/`.

[51] *Pomegranate*. `https://github.com/jmschrei/pomegranate`.