UiA University of Agder

An Environment-Adaptive Approach for Indoor Localization Using the Tsetlin Machine

ROBIN OLSSON OMSLANDSETER

SUPERVISORS

Lei Jiao
Ole-Christoffer Granmo

Master's Thesis
University of Agder, 2021
Faculty of Engineering and Science
Department of Information and Communication Technology

UiA
University of Agder
Master's thesis

Faculty of Engineering and Science
Department of Information and
Communication Technology

Abstract

Indoor positioning is a challenging task due to the small scale of area and the complex electromagnetic environment. Among different distance measurement schemes, Received Signal Strength Indication (RSSI) readings are commonly used in proximity and localization applications such as in BLE and Wi-Fi, because of the low power consumption and simplicity of retrieving this information. There are several approaches for RSSI based indoor localization, among which the deep-learning based models trained with fingerprinting data can achieve far superior localization accuracy compared with orthodox approaches, such as trilateration. However, fingerprinting requires extensive manual labor during the offline data collecting phase for training and cannot adapt well to changes in the environment.

In this thesis, we propose a novel environment-adaptive approach for indoor localization, which utilizes beacon-to-beacon RSSI readings that can be dynamically obtained directly from a WSN. The advantage of the beacon-to-beacon solution is that the offline training data can be obtained with minimal manual effort and can adapt to changes in the environment on the fly. In addition, for the AI algorithm, we propose a novel Tsetlin Machine (TM)-based approach for indoor localization. TM is a recently proposed powerful and energy efficient AI technique, which is particularly suitable for battery-driven devices. To study the feasibility and evaluate the system's performance, we set up a test bed in an office environment from which an extensive retrieval of RSSI readings were made and processed. The numerical results demonstrated that the proposed approach surpassed orthodox localization algorithms. Although the accuracy of the proposed approach is slightly lower than the WCL algorithm and the state-of-the-art AI based fingerprinting method, the advantages of the hardware friendliness, the unnecessity of involvement of end-users, and the adaptive property make it highly promising to the field of indoor localization.

Preface and Acknowledgements

This thesis was written in cooperation with Meshtech AS. I want to make a huge thanks to Meshtech AS, who provided me with the required tools, equipment, and BLE devices used in this thesis. A special thanks to Torjus Færsnes, who proposed the thesis definition. Thanks to Eirik Aanonsen for explaining the theory and providing research material on RSSI based localization. Thanks to Eivind Kristoffer Holst-Larsen for the help with configuring the network and placing the beacons. Thanks to Kirill Kovalenko, who provided me with the source code based on prior research at Meshtech AS.

Thanks to Lei Jiao and Ole-Christoffer Granmo for the excellent academic supervision. Thanks to Rohan Kumar Yadav for helping with the initial data collection. Thanks to Darshana Abeyrathna for the instructions on how I could train the RTM correctly. Thanks to Rebekka Olsson Omslandseter for the academic support.

I would also like to thank everyone mentioned for their invaluable guidance and discussions while working with this thesis. Without the help from all of you, we would not have achieved as valuable and interesting results as we have. Thank you all!

Table of Contents

Glossary

ADC Analog-to-Digital Converter.

AI Artificial Intelligence.

AIoT Artificial Intelligence of Things.

ANN Artificial Neural Network.

AoA Angle of Arrival.

AP Access Point.

BC Beacon.

BC-to-BC Beacon-to-Beacon.

BLE Bluetooth Low Energy.

BS Base Station.

CNN Convolutional Neural Networks.

DL Deep Learning.

FP Fingerprinting.

GAP Generic Access Profile.

GATT Generic Attribute Profile.

GPS Global Positioning System.

GW Gateway.

ILS Indoor Localization System.

IoT Internet of Things.

IP Internet Protocol.

LA Learning Automaton.

LS Least Squares.

LSTM Long Short-Term Memory.

MCTM Multi-Class Tsetlin Machine.

MCU Micro Controller Unit.

ML Machine Learning.

MS Mobile Station.

NFC Near-Field Communication.

OLM Organic Landmarks.

PSO Particle Swarm Optimization.

RF Radio Frequency.

RMSD Root-Mean-Square Deviation.

RNN Recurrent Neural Networks.

RP Reference Point.

RSSI Received Signal Strength Indication.

RTM Regression Tsetlin Machine.

SNR Signal-to-Noise Ratio.

SoC System-on-Chip.

SVM Support-Vector Machines.

TA Tsetlin Automaton.

TDoA Time Difference of Arrival.

TM Tsetlin Machine.

ToA Time of Arrival.

W-MCTM Weighted Multi-Class Tsetlin Machine.

W-RTM Weighted Regression Tsetlin Machine.

W-TM Weighted Tsetlin Machine.

WCL Weighted Centroid Localization.

WLAN Wireless Local Area Network.

WSN Wireless Sensor Network.

List of Figures

List of Tables

Table of Notations

Notation	Description
d	Distance
χ	Coordinate ($\chi = (x, y)$)
φ	Signal strength
c	Calibrated RSSI
t	Time
τ	Sampling interval
b	Binary literal ($b \in \{0, 1\}$)
η	Path loss exponent ($\eta \in [1, 4]$)
M	Number of access points
R	Number of reference points
N	Number of data points
B	Number of bits per feature
A	Gaussian distributed random variable
W	Rolling window length
L	Number of literals
E	Number of epochs
μ	Mean
σ	Standard deviation
s	TM precision
T	TM target value
C	Single clause value ($C \in \{0, 1\}$)
m	Number of clauses
X	Input vector
Y	Output vector
g	WCL attraction factor

Part I

Research Overview

Chapter 1

Introduction

Meshtech AS is a Norwegian technology company located in Arendal that develops full end-to-end IoT systems with their proprietary hardware and software solutions. Meshtech delivers their systems and services to various clients all around Norway. IoT devices and IoT technology are becoming increasingly popular, and the expanded use leads to the desire for improved functionality in connection to the devices and systems that Meshtech delivers. Localization technology associated with their devices is an interesting extension in this context. Therefore, Meshtech wants to investigate and start the development of their own localization system that can be deployed along with their existing hardware and in their proprietary BLE networks. Consequently, Meshtech proposed a master's thesis task to start this initiative. The task concerned developing a localization technique that required limited amounts of human intervention and could be used within Meshtech's systems. This thesis presents the methodology and results of the work related to solving this task.

Localization based on wireless signals is an interesting field of research. For outdoor localization problems, Global Positioning System (GPS) signals can provide high positional accuracy with approximately $5m$ deviation from the true position in open sky settings [1]. However, GPS and cellular systems fail to provide sufficient localization accuracy in indoor environments and in scenarios where the satellite or cellular signal is broken [2]. Consequently, GPS localization is considered to be an adequate method for localization in outdoor settings. In contrast, indoor settings are an ongoing development area because, e.g., GPS solutions' accuracy in these settings remains insufficient.

Received Signal Strength Indication (RSSI) is a quantity that is commonly used for proximity and localization estimation in Wireless Local Area Network (WLAN) technologies, such as, e.g., in Bluetooth Low Energy (BLE) and Wi-Fi networks. However, RSSI is not a standardized unit of measurement. RSSI, which is an indication of signal strength, remains a popular measure because RSSI readings can be obtained at a low cost through energy-efficient techniques that are easy to implement. RSSI strengths are highly influenced by environmental factors, such as absorption, interference from other objects, and diffraction. Therefore, solving localization problems based on these quantities can be a rather complicated challenge.

Over the last decade, the use of Machine Learning (ML) algorithms to solve various problems have become increasingly popular, and the rather computationally heavy Artificial Neural Network (ANN) and Deep Learning (DL) ANN have become the go-to tools for solving regression, classification, and audio- or image-processing problems. However, these methods often receive criticism because it is difficult to know the exact reasons behind the various outcomes the models arrive at, and they are often referred to as black boxes. At the same time, there exists models that have better interpretability, such as the Tsetlin Machine (TM). The TM has even achieved better results than ANNs for some datasets [3].

In recent years, Internet of Things (IoT) applications have gained vast attention from a wide range of industries [4]. And even more recently, it has been anticipated that ML and Artificial Intelligence of Things (AIoT) will transform the IoT industry. By using ML to perform data analysis directly on the edge, rather than in external cloud resources, we can effectively reduce the amount of data flowing in the network. In this way, we can reduce the bandwidth requirement, increase the range of the IoT devices and at the same time reduce the overall devices' power consumption. With AIoT technologies, such as TinyML, we are likely to see better utilization of the computational power available on modern Micro Controller Unit (MCU)s.

In this thesis, we will combine an innovative ML technique with RSSI measurements from IoT devices to make estimates of objects' location in indoor environments. Unlike the black box ML methods, such as DL ANN, we propose to use a novel TM approach. The TM is highly interpretable and provides competitive performance to other state-of-the-art ML algorithms. In addition, the TM can be implemented on small IoT units in a processing-efficient way and extensive research is initiated in this area [5]. In this way, our proposed method provides a good solution concerning the objectives of the thesis.

1.1 Motivation

Let us consider a BLE network deployed in a multi-floor hospital environment, where there is one beacon device placed in each room. A patient is wearing a wristband device with an alarm button. The system administrators want to know which floor and room the patient is located in when the patient pressed the alarm button. If we had a system that could automatically determine the patient's location, the system would provide potentially life-saving information for emergency staff at the hospital. With RSSI based indoor localization, automatic localization would be possible. The beacon devices would obtain RSSI readings from the wristbands, and we can implement methods for localization in association with this information.

For RSSI-based indoor localization, the target node, which we want to locate, transmits a signal or some data which is picked up by nearby listening Access Points (APs)[1]. The APs have fixed and known positions, while the location of the target nodes are traditionally derived using mathematical approaches, such as trilateration.

ML can be applied when processing the measured RSSI to improve the localization accuracy. Especially in WSNs with multiple anchor nodes, as ML algorithms, such as the ANN, can see relations between the data features which are not immediately apparent to us humans. The Tsetlin Machine (TM) is a relatively new but highly promising ML algorithm [3]. The TM has not been used for indoor localization yet. One potential with the TM, which we'll discuss further in this thesis, is that pre-trained TMs can be constructed using logic gates. In this way, trained TM models can potentially be constructed as individual electrical circuits and be placed directly on the System-on-Chip (SoC), providing ML with close to immediate prediction time on low-power embedded devices, which is a big advantage in the IoT systems that our solution is proposed for.

Meshtech's beacon devices features a *pinging* functionality. This functionality allows the system administrator to initiate advertisement on the beacon for a specified time duration. The advertisement packets are then picked up by other nearby listening beacons- or GW devices. The idea was that

[1]The terms Access Point (AP), Base Station (BS) and anchor nodes are used interchangeably in the literature. In this thesis, APs refer to both gateway and beacon devices with fixed positions. The terms blind node, target node and Mobile Station (MS) are also used interchangeably. In this thesis, these terms refer to the nodes that we want to locate.

this beacon-to-beacon data could be utilized for the purpose of indoor localization. This concept had not been thoroughly investigated prior to this thesis. This thesis will, thus, give Meshtech in-depth knowledge of how this functionality can actually be used and how well it possibly works.

1.2 Problem Statement of the Thesis

Fingerprinting is one of today's most favored approaches for indoor localization. However, fingerprinting is known to require a significant amount of pre-deployment efforts upfront [6]. Therefore, the method is associated with a high cost, which might not be ideal for all industrial-level environments, such as in, e.g., hospitals.

Although there has been prior research on how to reduce the pre-deployment effort of indoor localization systems, we explore if we can do this by using beacon-to-beacon information, which we can conveniently obtain directly from the WSN. The problem statement of the thesis is formulated as follows: Is it possible to capture and utilize in-network RSSI between beacons to build indoor localization models, reducing the amount of pre-deployment effort and still maintaining an acceptable localization accuracy? If beacon-to-beacon RSSI can be used for this purpose, then we would effectively have a more environment-adaptive approach for indoor localization.

In this thesis, we want to analyze the performance of the TM algorithm for solving indoor localization tasks. Although, this algorithm has shown great potential in other applications, with competitive accuracy compared with other conventional ML algorithms, such as ANNs [3], it has not been tested on localization problems before. Therefore, the use of TM in this thesis for indoor localization is novel in itself.

1.3 Objectives of the Thesis

The main objectives of this thesis can be summarized as follows:

- Obtain a RSSI-based dataset for indoor localization based on a real-world WSN test bed within an office environment using Meshtech proprietary BLE devices.

- Design an environment-adaptive approach for indoor localization by utilizing the beacon-to-beacon pinging data available in the Meshtech's BLE-based WSNs.
- Validate the beacon-to-beacon approach and compare its resulting localization accuracy with the state-of-the-art fingerprinting approach.
- Validate the performance of the TM model with the captured localization data and compare the results with other ML algorithms.

1.4 Contributions

The contributions of the thesis can be summarized as follows:

- We propose a unique method for transforming in-network beacon-to-beacon RSSI values into fingerprinting data. This data can be used for training indoor localization models and is, thus, a newly proposed method for constructing environment-adaptive Indoor Localization System (ILS)s.
- Our results demonstrated that our approach for transforming in-network beacon-to-beacon RSSI values provides improved localization accuracy compared to other traditional mathematical techniques, in cases where labor-intensive manual on-site fingerprinting is undesirable.
- We propose using the Regression Tsetlin Machine (RTM) for adjusting RSSI readings from beacons to enhance the localization accuracy of mathematical localization algorithms, which is a novel approach to the field of indoor localization.
- Additionally, we suggest using the Multi-Class Tsetlin Machine (MCTM) for solving indoor localization as a classification problem.
- The TM was compared with other popular ML algorithms, and our results demonstrated that the MCTM performed better than both the ANN and the random forest classifier utilized for comparison in this thesis.

There have been prior attempts to reduce the pre-deployment effort of indoor localization. However, our beacon-to-beacon approach is a unique way

of tackling this problem. Our approach is also specifically designed for industrial-level BLE networks and systems provided by Meshtech. It provides an alternative method of tackling localization in environments where manual on-site data surveys are undesired, and ultra-high-level localization precision is unnecessary. Our efforts to validate the performance of the TM for indoor localization is also a contribution to the existing literature. To our knowledge, this thesis presents the first reported TM approach for solving indoor localization problems.

1.5 Outline of the Thesis

This thesis is structured as follows:

Chapter 2 covers the underlying background theory and state-of-the-art in the field of RSSI-based indoor localization. Additionally, we present different methods for indoor localization and describe the TM algorithm utilized in this thesis.

Chapter 3 presents the proposed solution for the environment-adaptive indoor localization approach which utilizes beacon-to-beacon RSSI readings.

Chapter 4 presents the proposed solutions on how the TM can be used to improve localization in WSNs. We propose two solutions. The first solution utilizes the RTM to process the raw RSSI values, where we can combine the result with mathematical formulas to obtain a single point prediction of the target node's location. The second implementation utilizes the MCTM to find an object's position as a classification problem, where we link the outputted class to a specific location in the environment.

Chapter 5 presents the results for the proposed environment-adaptive beacon-to-beacon approach for indoor localization. Here, we compare the beacon-to-beacon approach with the well-established fingerprinting approach. Additionally, we analyze the performance of the proposed TM implementations and present a comparison between these results and the results obtained with other ML algorithms.

We conclude the thesis in **Chapter 6** and discuss how we can further enhance the proposed solutions in future work.

Chapter 2

Background and the State of the Art

In this chapter, we outline some of the state-of-the-art approaches and present related theory within the field of indoor localization. In addition, we explain the machine learning algorithms used in this project, such as, for example, the TM and the ANN.

We emphasize that both the terms positioning and localization are used interchangeably in the literature. However, in this thesis, we refer to positioning as the physical placement of the nodes in the environment, while localization refers to predicting the nodes' actual coordinates.

2.1 Indoor Localization

Indoor localization has witnessed an increased interest in recent years [7]. Fingerprinting using Wi-Fi and a mobile phone is expected to be one of the most popular methods for indoor localization, and it already achieves around $2 - 3m$ localization accuracy [8]. It has been reported that a localization accuracy of approximately $2m$ can be obtained with Wi-Fi and BLE based localization approaches [9]. In the WLAN segment, RSSI-based localization approaches, such as the fingerprinting method, are prominent because it requires no extra equipment [2]. In addition, other indoor lo-

calization methodologies, such as the Angle of Arrival (AoA), the ToA and the TDoA have been reported [10]. According to Bluetooth SIG, Bluetooth Direction Finding is expected to elevate the next generation of Bluetooth localization services [11].

2.1.1 Related Work

As we mentioned in the introduction, localization is an extensive area of research, and in what follows, we will summarize the most important research contributions within its paradigm. The reader is referred to [12], which presents a survey of different indoor localization systems, technologies, and related accuracy for additional details.

The authors in [13], propose an indoor localization algorithm based on the so-called Improved RSSI Distance Model. The proposed algorithm utilizes a Kalman[1] filtering technique to smooth out the fluctuations of the real-time RSSI readings. Furthermore, the proposed algorithm extracts an RSSI correction offset in real-time from existing APs in the network. It utilizes this offset to correct the RSSI readings from the target nodes, reducing the localization error. The proposed algorithm also uses a back propagation ANN with Particle Swarm Optimization (PSO) in order to convert the RSSIs into distances. Finally, the Least Squares (LS) algorithm is applied to predict the final position of the target node.

Chintalapudi et al. [6] propose a strategy for eliminating the on-site survey required in fingerprinting. The authors suggest that the target nodes themselves measure the RSSI of APs and report back these measurements when the location of the smartphone can be known through other means, such as via GPS signals. This approach reportedly yields a median of $2 - 7m$ localization error.

Wu et al. [14] propose a Wi-Fi-based positioning system named WILL, which does not require an on-site survey or prior knowledge of AP positions. Here the floor map of the entire building is constructed virtually, while Wi-Fi fingerprints and user movement are used together to predict the target node‘ location. This method reportedly achieves an average room-level accuracy of 86% among 16 rooms with $M = 26$ installed APs in an office building.

[1]Kalman filters are widely used in real-time applications which tracks moving objects, but will not be further explained in this thesis.

Wang et al. [15] propose an unconventional approach for indoor localization called UnLoc. This method utilizes other environmental Organic Landmarks (OLM) signatures, such as magnetic fluctuations in addition to typical RSSI readings together with data from gyroscopes and compasses. The data is used in an unsupervised learning environment, and the approach reportedly achieves $1 - 2m$ localization accuracy.

2.1.2 Received Signal Strength Indication

RSSI represents a measurement of the power level of a RF signal at a receiver's antenna. A RSSI reading is done using an ADC at the time a data packet arrives, and the power of the signal is thus measured. The strength of the received signal depends on the receiver's distance from the sender and the sender's broadcasting power and receiver's antenna power. However, RSSI is known to fluctuate significantly as it is influenced by many environmental factors, such as multipath propagation, reflection and signal fading [16]. For instance, the human body is known to absorb as much as $9dB$ of $2.4GHz$ Radio Frequency (RF) signals in free space environments [8]. Experiments in [17] show that the RSSI may drop $30dBm$ if the target node is moving toward the AP across $10cm$.

There is no standardization that links RSSI to a physical parameter. Consequently, different vendors and manufacturers use different scales and granularity when referring to RSSI. However, we emphasize that Decibel-Milliwatt (dBm) is a more typical unit of measurement in the literature. Meshtech's BLE devices are equipped with Nordic Semiconductor SoCs. Therefore, Meshtech's devices inherit the RSSI definition featured by Nordic Semiconductor chips, where RSSI is represented as an eight-bit long signed integer value that can be converted to dBm. The minimum RSSI provided by Meshtech devices is $-100dBm$, e.g., $\varphi_{min} = -100dBm$.

2.1.3 Indoor Radio-Channel Path Loss Model

Based on statistical analysis, it has been demonstrated that the channel fading characteristic of RF signals follow a lognormal distribution [13]. The path loss model for indoor RF signals is a widely used formula linking the path loss of RF signals to distance [18]. According to this model, the signal

path loss, or RSSI φ, given in decibels-milliwatts, at a distance d can be obtained with the following formula:

$$\varphi(d)\ [dBm] = \varphi(\bar{d}_0) - 10\eta \log_{10}\left(\frac{d}{d_0}\right) + A_\sigma, \tag{2.1}$$

where $\varphi(\bar{d}_0)$ is the average signal path loss at a specific reference distance (normally $d_0 = 1m$), η is an environment-dependent path loss exponent and A_σ is a Gaussian random variable with zero mean and σ standard deviation in dBm [19].

The path loss exponent η determines the rate at which the signal decays [18]. The value of η is increase in environments with more obstacles [13], and $\eta = 2$ represents a free-space environment [20]. η is usually a value between one and four, i.e., $\eta \in [1, 4]$. Because $\eta = 2$ represents a free-space environment, η is usually higher than two.

$\varphi(\bar{d}_0)$ is often referred to as the calibrated RSSI, or one-meter-RSSI (as d_0 is commonly and conveniently set to $1m$). The one-meter-RSSI represents the average RSSI granted at a one-meter distance. This value is determined by the antenna characteristics of both the sender and the receiver [20]. Thus, $\varphi(\bar{d}_0)$ is generally a factory-calibrated constant provided by the product manufacturer.

By conveniently setting $d_0 = 1m$, renaming $\varphi(d_0)\ |_{d_0=1m}$ to c, and considering the average RSSI $\bar{\varphi}$ (thus, removing the Gaussian noise A_σ), the formula can be further simplified to:

$$\bar{\varphi} = c - 10\eta \log_{10} d. \tag{2.2}$$

By reversing Equation 2.2, the distance d can be obtained as:

$$d = 10^{\frac{c-\bar{\varphi}}{10\eta}}. \tag{2.3}$$

The reader should note that it can be challenging to obtain accurate distances, especially if the object is moving in real-time, even when a sliding window is applied to the RSSI. According to Equation 2.1, the signal

strength is weakened when the distance d is increased. At the same time, the Gaussian random variable is unaffected by distance. Based on this, we inherently know from Equation 2.1 that the Signal-to-Noise Ratio (SNR) decreases as the distance increases.

2.1.4 Trilateration

Trilateration is a well-known and classical geometrical approach for indoor localization. It is a method to determine an target node's coordinates by using simultaneous range measurements from three APs positioned at known sites, as shown in Figure 2.1 [21]. The intersection between the three circular ranges, corresponds with the target node‘s coordinates. When considering two Cartesian dimensions, the trilateration problem can be expressed with the three following equations:

$$\begin{cases} (x_1 - x)^2 + (y_1 - y)^2 = d_1^2, \\ (x_2 - x)^2 + (y_2 - y)^2 = d_2^2, \\ (x_3 - x)^2 + (y_3 - y)^2 = d_3^2, \end{cases} \tag{2.4}$$

where we denote the position of the target node as $\chi = (x, y)$ with x and y representing its coordinates. Further, (x_j, y_j) are the coordinates of the j^{th} AP ($j = \{1, 2, 3\}$) and d_j are the distances between the j^{th} AP and the target node. For RSSI-based localization in WSNs, the distances d_i are commonly calculated by reversing the propagation model, as shown in Equation 2.3. Solving the set of equations above in terms of χ obtains an estimate of the target node‘ position. In [10], a novel solution for the trilateration problem is presented. The target node‘ location can be obtained as:

$$\chi = \begin{cases} x = \frac{a - y(y_3 - y_2)}{x_3 - x_2}, \\ \\ y = \frac{b(x_3 - x_2) - a(x_1 - x_2)}{(y_1 - y_2)(x_3 - x_2) - (y_3 - y_2)(x_1 - x_2)}, \end{cases} \tag{2.5}$$

where the variables a and b can be obtained by:

$$a = \frac{(d_2^2 - d_3^2) - (x_2^2 - x_3^2) - (y_2^2 - y_3^2)}{2}, \tag{2.6}$$

and:

$$b = \frac{(d_2^2 - d_1^2) - (x_2^2 - {}_1^2) - (y_2^2 - y_1^2)}{2}, \tag{2.7}$$

respectively. This approach will obtain the precise coordinates of the target node, as long as the distances d_j are accurate. However, the method above is only applicable in scenarios with $M = 3$ APs. Thus, three APs are needed in order to determine the target node's coordinates on a 2D plane. If we instead consider three spheres, we would get two ambiguous points in the z-dimension. Trilateration may be considered an outdated localization approach today [19], but it still maintains some significance within the field of indoor localization, as it is a well-known algorithm. The distance between two known coordinates, $\chi_1 = (x_1, y_2)$ and $\chi_2 = (x_2, y_2)$, can be computed using the Pythagorean theorem as follows:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{2.8}$$

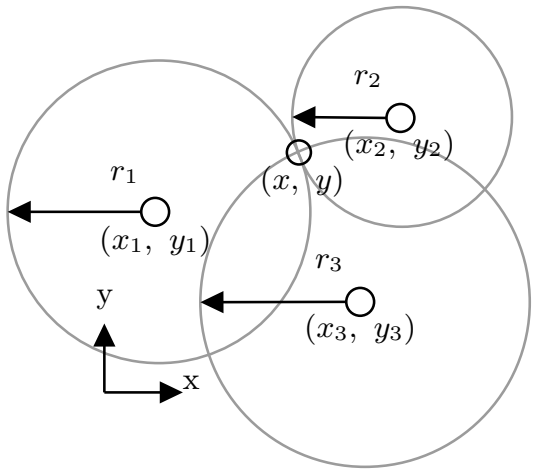


Figure 2.1: The trilateration problem.

2.1.5 Min-Max Localization Algorithm

The min-max localization algorithm is a popular localization approach [22]. This approach uses rectangular shapes rather than circles, as shown in Figure 2.2. The inner-most intersection determines the current prediction of the node's location. In [22], they utilize an approach where the area marked by the min-max algorithm evaporates and diffuses for each time step. The marks evaporate slower than new marks occur, and all existing marks are summed up. Thus, if the mobile node stays still in one place, its current position will be reinforced over time.

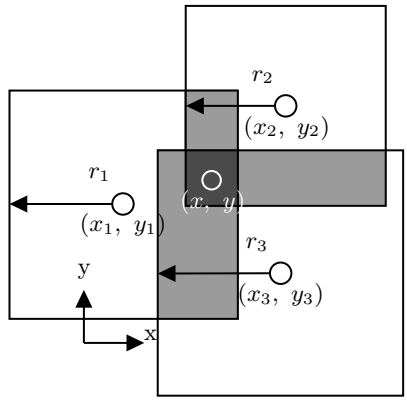


Figure 2.2: The min-max localization algorithm.

2.1.6 Modified Weighted Centroid Localization Algorithm

An estimate of the target node‘ coordinates can be found with the Traditional Centroid Algorithm, which returns the center coordinate between visible APs. However, the Weighted Centroid Localization (WCL) algorithm, which adds weights to the distances d_j, is normally preferred for indoor localization as it provides more accurate estimates [23]. In simple terms, the WCL algorithm predicts the node's coordinates by summing up the coordinates of all visible APs and weighting them according to their estimated distances. In [24], an exponent g is introduced on the weights, in order to construct a so-called "attraction-field" around the target node, so that the nearest APs weighs more for determining the estimate. Consequently, this estimation technique for finding the target node‘ coordinates

can be found with:

$$\hat{\chi} = \begin{cases} x = \frac{\sum_{i=1}^{M}(\hat{r}_i^{-g} x_i)}{\sum_{i=1}^{M}(\hat{r}_i^{-g})} \\ \\ y = \frac{\sum_{i=1}^{M}(\hat{r}_i^{-g} y_i)}{\sum_{i=1}^{M}(\hat{r}_i^{-g})}, \end{cases} \tag{2.9}$$

where the predicted position $\hat{\chi}$ becomes the center coordinate in respect to all M APs when $g = 0$, and an increasing value of g increases the relative weight of the nearest APs by reducing the weight of APs further away. When combining the WCL algorithm with fingerprinting, it has been stated that the required number of reference points can be reduced by up to 40% compared with the typical fingerprinting approach [25]. WCL is relatively easy to apply, even in cases with more than three APs, but may only grant a rough estimate of the target node‘ actual location, even when the exact distances d_i are known. The WCL algorithm can, however, work well in practical situations.

2.1.7 Least Squares Algorithm

The Least Squares (LS) algorithm is a standard approach for solving regression problems in overdetermined systems [13]. This approach can be used to estimate the location of a target node more accurately than the triangular centroid algorithm. For systems with $M \geq 3$ APs, the following set of equations can be obtained:

$$\begin{cases} (x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2 = d_1^2, \\ (x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2 = d_2^2, \\ \vdots \\ (x_M - x)^2 + (y_M - y)^2 + (z_M - z)^2 = d_M^2 \end{cases} . \tag{2.10}$$

Using LS, equation M is subtracted from all the first $M - 1$ equations [13]. Further, the following linear expression can be obtained:

$$\alpha\chi = \beta, \tag{2.11}$$

where α is given by:

$$\alpha = \begin{pmatrix} 2(x_1 - x_M) & 2(y_1 - y_M) & 2(z_1 - z_M) \\ 2(x_2 - x_M) & 2(y_2 - y_M) & 2(z_2 - z_M) \\ \vdots & \vdots & \vdots \\ 2(x_{M-1} - x_M) & 2(y_{M-1} - y_M) & 2(z_{M-1} - z_M) \end{pmatrix} \tag{2.12}$$

and β is given by:

$$\beta = \begin{pmatrix} x_1^2 - x_M^2 + y_1^2 - y_M^2 + z_1^2 - z_M^2 + d_1^2 - d_M^2 \\ x_2^2 - x_M^2 + y_2^2 - y_M^2 + z_2^2 - z_M^2 + d_2^2 - d_M^2 \\ \vdots \\ x_{M-1}^2 - x_{M-1}^2 + y_{M-1}^2 - y_M^2 + z_{M-1}^2 - z_M^2 + d_{M-1}^2 - d_M^2 \end{pmatrix}. \tag{2.13}$$

Finally, the target node‘ coordinates χ can be obtained as:

$$\chi = (\alpha^T \alpha)^{-1} \alpha^T \beta, \tag{2.14}$$

where α^T is the transposed version of matrix α. Like trilateration, LS provides the exact coordinates of the target node as long as the distances d_i are correct. One major difference between this method and trilateration, is that LS handles $M \geq 3$ APs. LS can also easily be used together with three Cartesian dimensions as long as the APs are not placed uniformly along any of the axes.

2.1.8 Fingerprinting

Fingerprinting is one of today's most favored approaches for indoor localization, because it generally provides improved localization accuracy compared to orthodox approaches, such as trilateration [26]. The fingerprinting approach involves two phases; (1) the offline phase and (2) the online phase. In the offline phase, normally, RSSI readings from all visible APs is captured at R reference points. Each sample, X_φ, consist of M readings, i.e., $X_\varphi = \{\varphi_1, \varphi_2, ..., \varphi_M\}$ and is linked to the reference points coordinates χ_r.

This concept is shown in Figure 2.3. The goal is to capture the fingerprint of all the reference points and then use this information to train a localization algorithm that adapts to the given environment. The fingerprinting data can for instance be used to optimize the parameters of the radio-channel propagation model presented in Section 2.1.3.

When the localization service is live, referred to as the online phase, various approaches can be used to determine the location of the target node. Fingerprinting can be solved as a classification problem, where a multi-class prediction model, e.g., a Bayes classifier, can be used to predict at which reference point the target node is currently positioned.

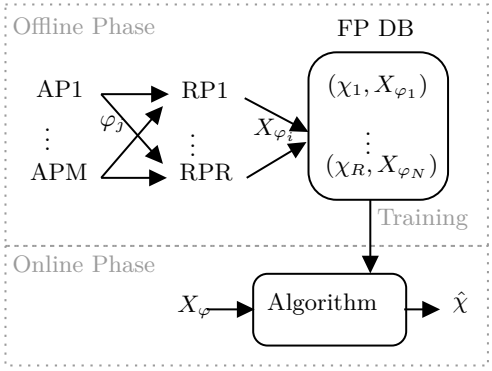


Figure 2.3: The fingerprinting localization approach.

2.2 Bluetooth Communication

Bluetooth operates on the 2.4GHz non-licensed ISM RF band, spanning from 2,400 MHz up to 2,483.5 MHz [27]. WLAN technologies, such as Wi-Fi, also operate within this ISM band. Bluetooth features adaptive hopping. Consequently, Bluetooth devices are less likely to interfere with other devices operating at the same frequencies, and are also less likely to be disturbed in-operation themselves. BLE uses 40 individual radio channels, where three of them (referred to as 37, 38 and 39) are used for advertisement packets sent via Generic Access Profile (GAP). The remaining 37 channels $(0 - 36)$ are used for data packets transmitted via Generic Attribute Profile (GATT). Figure 2.4 demonstrates the different frequency channels in Bluetooth systems.

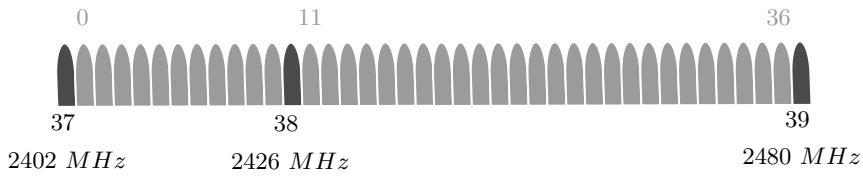


Figure 2.4: BLE radio frequency channels.

When a device is advertising, it is considered to be a *peripheral device*. The peripheral device can only send advertisement data on one channel at a time. In the meantime, a listening device, referred to as the *central device*, will only listen to one channel at a time. In addition, the listener might only be listening for at specific time intervals, determined by a so-called *scan window*. For that reason, the advertisement packet will only be picked up by a listening device with a certain probability.

The central device can initiate a connection with the peripheral device. This is achieved by an agreement of certain connection parameters, such as a certain hopping sequence, using the GAP protocol, before switching over to GATT once the connection has been established. The agreed-upon hopping sequence among the 37 remaining frequency channels is utilized for transmitting data packets when connected. Together, the two devices, from now on referred to as the master and the slave, form a so-called piconet. The master device can connect with multiple slaves, but each piconet can only have one master. However, the master can operate as a slave in another piconet, which is the case for Meshtech's beacon devices.

2.2.1 Bluetooth Networking

Meshtech has constructed their own proprietary networking protocol extending the traditional BLE protocol. This protocol, referred to as the Meshtech network protocol, features network trees and routed messages. Here, terms such as gateway and beacons emerge. In the world of IoT systems, the gateway act as an entry point between two different communication protocols. In the sense of Meshtech's protocol, the gateway act as an interface between IP based communication and the BLE network. Meshtech devices can form a so-called Network Tree, which is shown in Figure 2.5.

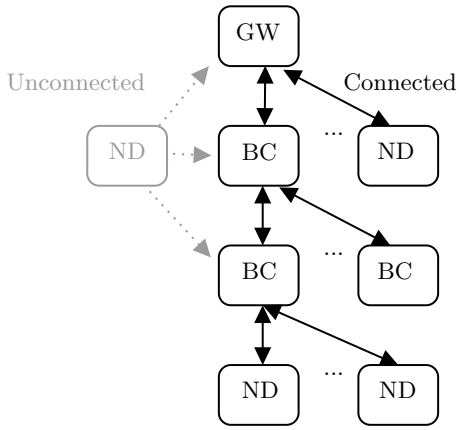


Figure 2.5: The Meshtech protocol network tree.

Each beacon can only have one parent node, but can have multiple child nodes. In terms of BLE, the Meshtech beacons act as slave and master at the same time, forming their own sub-piconets. The reader should note that multiple beacon devices can be connected to the same parent. In practice the network tree can become fairly large, with many parent and child node relations, potentially covering a vast physical space. The Meshtech protocol features routing, which means that messages sent from the GW can be routed to specific nodes or beacons further down in the network tree. At the same time, messages sent from edge nodes can climb the network tree and arrive at the GW. Edge nodes may operate in either connected or unconnected mode, as is illustrated in the figure. Packets sent from unconnected node may be picked up by any nearby listening AP.

2.3 Machine Learning

Artificial Intelligence (AI) is the paradigm of building computer algorithms that mimic human or animal behavior. Today, ML plays a significant role in the AI domain. ML involves building computer algorithms that can learn to perform specific actions or process data, given a set of inputs.

2.3.1 Artificial Neural Networks

Artificial Neural Network (ANN) is an approach within the field of ML, where the computer algorithm tries to imitate the processes in an organic brain in order to solve complex problems [28]. ANNs can be used to solve classification, regression, text generation, and image- or audio processing problems. ANNs are constructed using artificial neurons, connected through weighted connections, similar to the synapses in the organic brain. The ANN can consist of several layers of neurons [29], as visualized in Figure 2.6. An ANN has one input layer and one output layer. ANNs can have an arbitrary number of hidden layers, where each additional neuron adds complexity to the model. Deep Learning (DL) was introduced in a paper by Geoffrey Hinton Et Al. in 2006 [30]. This paper revived the scientific community's interest in ANN, as DL could solve large-scale complex problems with state-of-the-art precision. The term DL is often used for highly complex ANNs with many hidden layers. When each neuron in the previous layer is connected to all the neurons in the next layer, the structure of the ANN is referred to as *dense*, and it is realized through a *feed-forward process*. In the feed-forward process, the output of a single neuron can be computed as:

$$y_i = f\left(b + \sum_{i=1}^{n} x_i, w_i\right), \tag{2.15}$$

where b is a bias, n is the number of neurons in the previous layer, x_i is the i^{th} previous layer neuron's output, w_i is the weight between the i^{th} previous layer neuron and the current neuron, and $f(z)$ is an activation function. Consequently, y_i can be fed as an input for the next layer's neurons. The value of all the neurons in the output layer represents the model's prediction $\hat{Y}$. Various activation functions can be used, such as ReLU, Soft-max, or linear activation. Linear activation is expressed as $f(z) = az$, where a is an arbitrary constant. ReLU activation is expressed with $f(z) = \max(0, z)$. Soft-max activation is commonly used in the final layer of classification models, as it transforms real values into probabilities [31]. Soft-max activation for the j^{th} neuron can be expressed with:

$$f_j(z) = \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}}, \tag{2.16}$$

where e is Euler's number, z is the input vector and $j \in \{1, 2, ..., n\}$.

During training, ANNs achieves a stochastic gradient descent through a process called backpropagation, which is a supervised learning strategy [29]. The weights and biases for each neuron are adjusted based on the error of the model's prediction, going backward, starting from the output layer. The error is calculated by comparing the model's prediction $\hat{Y}$ with the target output Y.

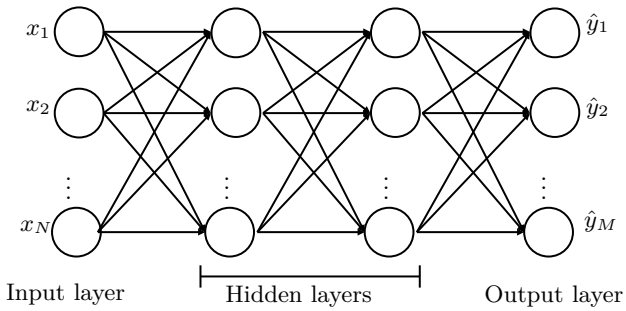


Figure 2.6: Layer structure of sequential ANNs.

Figure 2.6 shows the general structure of classical densely connected ANN. However, more complex model structures exist, such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN). These schemes will not be explained in depth in this thesis. The general principle behind RNNs is that the output from a previous time step is used as input in the next time step. Thus, the RNN can be used for time-series data, and in the case of indoor localization, the information of earlier positions can be beneficial for tracking moving objects.

2.3.2 Embedded Machine Learning

ML processed physically on embedded devices is a relatively new field of research, which has gained much attraction recently because of the growth within both the ML- and IoT paradigms. TinyML is a paradigm that aims to integrate deep learning on small and ultra-low-power embedded devices [32]. This is especially relevant for the TM approach presented in the following chapter, because it has a particularly promising potential in regards to processing within IoT hardware [5].

2.4 The Tsetlin Machine

The Tsetlin Machine (TM) is a relatively new and promising ML algorithm, that utilizes propositional logic in order to solve large-scale and complex pattern recognition problems [3]. The TM is computationally simple and easy to interpret [33] because it utilizes the Tsetlin Automaton (TA) as its fundamental building block. In benchmarks, the TM has provided competitive accuracy compared to other state-of-the-art ML algorithms, such as Support-Vector Machines (SVM)s, Decision Trees, Random Forest, Naive Bayes classifiers, Logistic Regression and Artificial Neural Network (ANN) [3]. It outperforms SVM, logistic regression and ANN in several benchmark datasets, such as the Iris Dataset and the MNIST dataset [33]. By the authors of [34], it was demonstrated that the TM converges after fewer epochs and is more memory- and power efficient than the popular ANN. Abeyrathna et al. showed in [35] that by using parallel and asynchronous clause evaluation during training, the TM could achieve up to 50 times faster learning than its predecessor.

2.4.1 Tsetlin Automaton

The TA is considered to be the very first Learning Automaton (LA) introduced by M. L. Tsetlin in the early 1960s [3]. It is a versatile and straightforward learning mechanism with rapid and accurate convergence and low computational complexity. The TA, referred to as the *learner*, can learn the optimal action in unknown stochastic scenarios through interaction with an Environment, referred to as the *teacher*. The teacher provides feedback to the learner in terms of penalties and rewards upon actions made/selected by the learner. Consequently, through the TA choosing actions, and getting feedback on them from the Environment, the automaton will converge to the action that gives it the least probability of penalty over time. This is true, even for environments that change over time. The TA is based on states, where its current state, ϕ_u, determines which action it selects. A TA for a two-action Environment is shown in Figure 2.7. The TA in Figure 2.7 has $2N$ states and 2 actions. Figure 2.7 shows how the TA switches between states based on the feedback from the environment. In groups of multiple TAs, more complex functionality can be performed, which is the case of the TM.

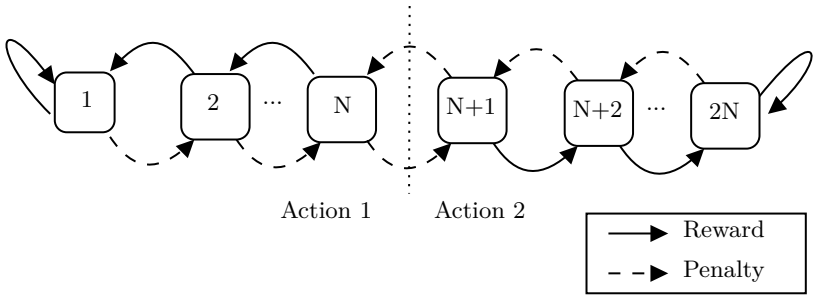


Figure 2.7: A Tsetlin automaton for two-action environments.

2.4.2 Classical Tsetlin Machine

The TM uses multiple two-state TAs, like the one shown in Figure 2.7 [33], that cooperates in complex learning schemes. The TAs represent the literals, which in groups form so-called conjunctive clauses. The inputs and outputs of the TM can be represented using bits. The main internal logic of the TM, such as recognition and training, can be done through manipulating those bits. This way, the TM naturally requires small computational resources and can work close to the hardware. In addition, the logic behind the TM's predictions becomes fully explainable. The output of the TM is based on the summation of evidence or votes from the clauses, where the majority of votes determines the output $\hat{y}$. This can be expressed as:

$$\hat{y} = u\left(\sum_{j=1}^{m/2} C_j^1(X) - \sum_{j=1}^{m/2} C_j^0(X)\right) \tag{2.17}$$

and

$$u(v) = \begin{cases} 1, & \text{if } v \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{2.18}$$

where C_j^1 denotes the positive polarity clauses, C_j^0 denotes the negative polarity clauses, X is the input vector and m is the total number of clauses [3]. Here, $u(v)$ is a binary step function, referred to as the threshold function. The structure of the TM is shown in Figure 2.8.

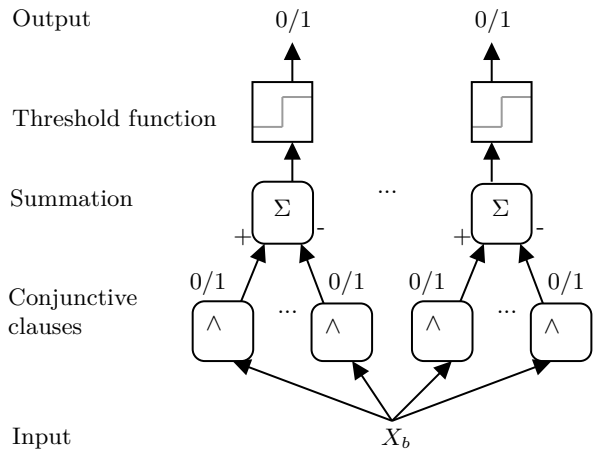


Figure 2.8: The Tsetlin Machine.

The TM learning algorithm is a reinforcement based learning algorithm [33]. It has two types of feedback, namely, Type I and Type II. Type I feedback eliminates false positives and reinforces true positives output, while type II feedback eliminates false positives output. The complete feedback probability table from [33] is shown in Table 2.1. The table shows how the probability for a reward, or penalty, is calculated for individual TAs. The reward and penalty probabilities depend on the clause's output, the TA's action, as well as its input literal. s is referred to as the precision.

Table 2.1: Tsetlin Machine feedback probability table.

Feedback Type			I				II			
Clause Output			1		0		1		0	
Literal Value			1	0	1	0	1	0	1	0
Current State	Include	Reward	(s-1)/s	NA	0	0	0	NA	0	0
		Inaction	1/s	NA	(s-1)/s	(s-1)/s	1	NA	1	1
		Penalty	0	NA	1/s	1/s	0	NA	0	0
	Exclude	Reward	0	1/s	1/s	1/s	0	0	0	0
		Inaction	1/s	(s-1)/s	(s-1)/s	(s-1)/s	1	0	1	1
		Penalty	(s-1)/s	0	0	0	0	1	0	0

2.4.3 Multi-Class Tsetlin Machine

The Multi-Class Tsetlin Machine (MCTM) can be used for pattern recognition problems, where the task is to predict one class among n classes, given an input pattern, X [3]. Here, the majority vote determines the outputted class. This is done by replacing the threshold function featured in the classical TM with a arg max operator. The class selection done by the MCTM can be expressed as:

$$\hat{y} = \arg\max_{i=1,\ldots,n} \left\{ \sum_{j=1}^{m/2} C_j^{1,i}(X) - \sum_{j=1}^{m/2} C_j^{0,i}(X) \right\}, \tag{2.19}$$

where the arg max operator returns the index i, which is assigned to a specific class. The MCTM uses the same feedback table as the classical TM during training. The logical structure of the MCTM is shown in Figure 2.9.

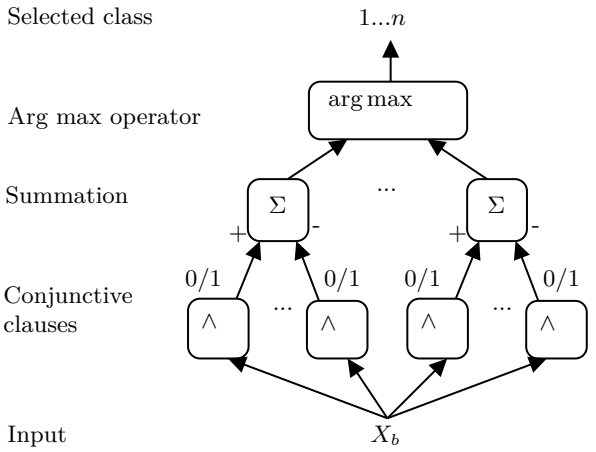


Figure 2.9: The Multi-Class Tsetlin Machine.

2.4.4 Regression Tsetlin Machine

The Regression Tsetlin Machine (RTM) was recently introduced to solve regression problems using the TM algorithm. The Regression Tsetlin Machine (RTM) transforms input patterns into a single continuous output and

can be used to solve regression problems where a single continuous value is desired [33]. In the RTM the polarity of the individual clauses is removed. Instead, the total vote count represents a single continuous value. The summation operator outputs a value between 0 and T, representing the number of clauses that evaluate to $'1'$. Hyper parameter T is referred to as the summation target [36]. The value outputted by the summation operator is then normalized to match the min- and max boundaries of the desired regression output. The output of the RTM can be obtained by:

$$\hat{y} = \frac{y_{max}}{T} \sum_{j=1}^{m} C_j(X), \tag{2.20}$$

where y_{max} is the maximum output value among all N training samples $Y = [y_1, ..., y_N]$, T represents the maximum number of votes, m is the number of clauses, $C_j(X)$ represents the output from the j^{th} clause given the input vector X. The RTM structural logic is shown in Figure 2.10. The RTM uses Type I- and Type II feedback, similar to the classical TM, but with different criteria as follows:

$$\text{Feedback} = \begin{cases} \text{Type I}, & \text{if } \hat{y} < y \\ \text{Type II}, & \text{if } \hat{y} > y \end{cases}, \tag{2.21}$$

where $\hat{y}$ is the prediction, and y is the target value. Type I feedback increases the number of clauses which evaluate to $'1'$ if the $\hat{y} < y$ while Type II feedback decreases the number of clauses which evaluate to $'1'$ This way, the loss is minimized over N training samples. A probability activation function stabilizes the learning. This function decreases the probability of giving feedback if the outputted value is close to the desired value, effectively reducing oscillation near the target value. The feedback probability p can be obtained using:

$$p = \frac{K(\hat{y} - y)}{y_{max}}, \tag{2.22}$$

where K is a constant.

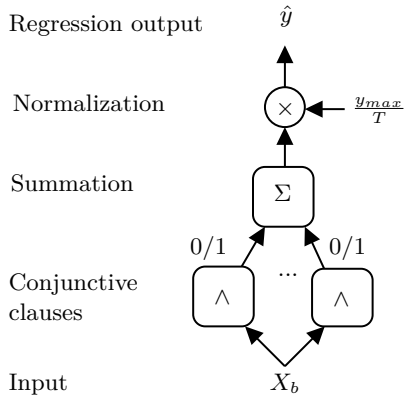


Figure 2.10: The Regression Tsetlin Machine.

2.4.5 Weighted Tsetlin Machine

In the classical Tsetlin Machine, every additional clause can contribute to improving the accuracy, but linearly adds computation time and memory usage [36]. The Weighted Tsetlin Machine (W-TM) aims to reduce the computation time and memory usage by applying weights to the individual clauses. Reportedly, the W-TM outperforms the regular TM when using the same number of clauses m [36]. Here a positive real-valued weight w_j is multiplied with the j^{th} clause's output:

$$\hat{y}'(X) = \sum_{j=1}^{m/2} w_j^+ C_j^+(X) - \sum_{j=1}^{m/2} w_j^- C_j^-(X), \tag{2.23}$$

where the output $\hat{y}'$ becomes real-valued. The step function from the classical TM in Equation 2.18, can be applied as follows:

$$\hat{y} = u(\hat{y}'(X)). \tag{2.24}$$

The weights can also be applied to the MCTM. Here the predicted class $\hat{y}$ is given with:

$$\hat{y} = \arg \max_{i=1,\ldots,m} \left\{ \hat{y}_i'(X) \right\}. \tag{2.25}$$

In general, the W-TM trains similarly to the classical TM, but the weights are also updated. When training starts, each weight w_j equals 1.0. For Type I feedback, each positive or negative clause evaluating '1' has its weight updated by multiplying with $(1 + \gamma)$, where γ is the learning rate and $\gamma \in [0, \infty)$. For Type II feedback, each positive or negative clause evaluating '1' has its weight updated by dividing by $(1 + \gamma)$. Clauses evaluating '0' keep their weights unchanged. This concept is realized by the following rules:

$$w_j = \begin{cases} w_j(1+\gamma), & \text{if Type I and } C_j(X) = 1 \\ \frac{w_j}{(1+\gamma)}, & \text{if Type II and } C_j(X) = 1 \\ w_j, & \text{otherwise.} \end{cases} \tag{2.26}$$

Part II

Contributions

Chapter 3

Utilizing Beacon-to-Beacon Data for Indoor Localization

In this chapter, we present our environment-adaptive approach for indoor localization. Our approach utilizes in-network beacon-to-beacon data to construct indoor localization models for BLE-based WSNs. While the gateway and beacon devices function as Access Point (AP)s with known positions in the WSNs, the edge nodes, such as wearable devices, are typically the ones we want to locate. The beacon-to-beacon data refers to RSSI measurements done in-between the beacons themselves, without involvement of any wearable devices, or other edge nodes. The beacon-to-beacon RSSI is measured whenever a beacon sends a advertisement messages which is picked up by other nearby listening beacons, tuned in on the same radio channel. The RSSI readings between each pair of beacons are then collected at the gateway. Our hypothesis is that the beacon-to-beacon data contains valuable information about the surrounding environment, and can, in the end, be used to increase the localization accuracy of the Indoor Localization System (ILS). This is opposed to simply guessing the environmental characteristics, or doing the tedious task of collecting fingerprinting data from nodes. In total, we can get up to $M!$ extra data features with this approach, and normally, this information remains unused in conventional ILSs. From this point forth, we refer to this approach as the BC-to-BC approach [1].

[1]Throughout this thesis, the term BC-to-BC refers to our approach of constructing localization models using beacon-to-beacon data. When we explicitly refer to the data, or the direction of the signal, we use the term beacon-to-beacon.

With the BC-to-BC approach, we propose a strategy to adapt beacon-to-beacon data into a node-to-beacon scenario, because the goal is to locate nodes, and not the beacons. The main idea is to train a fingerprinting-based localization algorithm. Each pinging beacon is considered an individual reference point. We propose methods in order to fill potentially missing data points and data features, specifically areas that are not covered by the beacon fingerprints. The resulting beacon fingerprints, and generated data points, are then combined in order to train a regression-based Machine Learning (ML) algorithm, that can process the RSSI and enhance the localization accuracy accordingly. Finally, a mathematical approach, such as trilateration or LS, can be applied to estimate the mobile nodes' location. With the BC-to-BC approach, re-calibration of the ILS can be done automatically by an application, and the approach can easily be adapted to various environments.

This chapter is structured as follows: Section 3.1 will discuss the critical challenges with the BC-to-BC approach. Section 3.2 will explain the different strategies for mitigating the challenges presented in the previous section. In addition, we will explain how an BC-to-BC-based ILS can be constructed.

3.1 Challenges

The goal of the BC-to-BC approach is to utilize beacon-to-beacon data in order to construct ILSs that functions well in localization of wearable devices, referred to as the target nodes. However, from the theory presented in Chapter 2.1.2, the beacon-to-beacon data can potentially be quite different from a node-to-beacon scenario, depending on the antenna characteristic of the different devices. Translating the beacon-to-beacon to a node-to-beacon scenario in a sensible manner, is one of the key challenges with the BC-to-BC approach. Additionally, if we also consider each beacon node to be placed at a reference point, only a handful, specifically M, reference points are provided by the beacon-to-beacon data. These M reference points may not be sufficient to train a ML algorithm, as we want the ML to recognize all areas of the room and not be limited to these points. Through the beacon-to-beacon data, the RSSI characteristic of other areas in the room remain unknown. Some of the challenges related to only using beacon-to-beacon data can be summarized as follows:

- From the beacon-to-beacon data, we get a maximum of M reference points, which is likely to be insufficient to train an accurate fingerprinting based localization model.

- Let's say $d_{BC \rightarrow BC_{min}}$ is the minimum distance between two beacon nodes in the environment. The mobile node can be positioned closer to one of the beacons than $d_{BC \rightarrow BC_{min}}$. However, the beacon-to-beacon data will not contain RSSI readings which covers distances shorter than $d_{BC \rightarrow BC_{min}}$. These data points must therefore be approximated.

- M input features are expected in the typical fingerprinting approach. However a pinging beacon node will not measure it's own signal strength. For each beacon node reference point, we will get a maximum of $M-1$ features, where the reading from the pinging beacon itself is missing.

- The beacon device and the mobile node may not share similar antenna characteristics, i.e., $c_{BC \rightarrow ND} \neq c_{ND \rightarrow BC}$. Thus, the Path Loss Model presented in Equation 2.1 can differ between the two scenarios.

3.2 Proposed Methods

In this section we present the proposed BC-to-BC method, and how to overcome the challenges presented in Section 3.1. The general methodology of the BC-to-BC approach is visualized in Figure 3.1. The figure illustrates the data flow in the system, from processing the raw beacon-to-beacon data, to training the ML model and running the resulting ILS. Here, $X_{BC \rightarrow BC_{Raw}}$ denotes the raw beacon-to-beacon data, which contains a complete list of all RSSI measurements done by the system during the data capture session. Each data point in $X_{BC \rightarrow BC_{Raw}}$ contains the measured RSSI, φ, the ID of the transmitting device, the ID of the receiving device and potentially a reference point ID linked to the current position transmitting node. The raw data is converted to typical fingerprinting data $X_{\varphi} = \{\varphi_1, \varphi_2, ..., \varphi_M\}$ using our proposed post-survey dataset compiler algorithm. The reader should note that the coordinates of all reference points, χ_r, and coordinates of the beacons χ_j should be contained in a separate file. We propose to use a ML algorithm, such as the RTM in order to convert the RSSI readings into adjusted distances, $\hat{Y}_d$, before applying the LS algorithm to compute the final estimation of the node's coordinates, where each distance $\hat{y}_j$ correlates

to RSSI reading φ_j. The individual components of the schematic will be further explained in the following sub-sections.

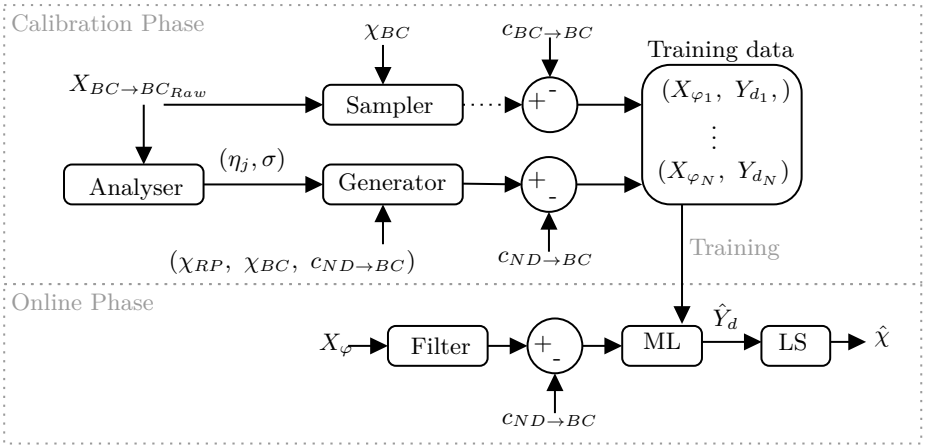


Figure 3.1: The BC-to-BC algorithm.

3.2.1 Post-Survey Dataset Compilation

As explained in Section 2.1.8, typical fingerprinting training datasets contain N samples with M RSSI measurements each. The fingerprinting dataset can contain a maximum of $N \times M$ individual RSSI readings. However, there is no guarantee that M readings are received for each exported sample, depending on where the broadcasting device is positioned. For instance, some APs can be out of reach (not visible). Some data packets can be lost due to congestion in the network. For this reason, constructing fingerprinting datasets can be a challenge in itself. Assuming that we cannot wait for M readings indefinitely, a sampling interval τ is introduced, where potentially missing data, has to be filled using interpolation techniques, such as padding. In terms of constructing a fingerprinting dataset which discards as few readings as possible, we suggest doing the sampling process after all raw data has been collected. By doing this, further analysis of the raw data is possible. Optimal fingerprinting parameters, such as the sampling interval τ and rolling window length W, can be derived before the final fingerprinting dataset is constructed. With this approach, it becomes easier to identify how many APs are in fact visible near the broadcasting device. Thus, we can wait for fewer readings before exporting a single sample. Consequently,

we can construct fingerprinting training data with maximum utilization of the available raw data.

For typical fingerprinting, the node must be manually positioned at the reference point, and the coordinates of the node must be noted down. However, for the BC-to-BC approach, the reference point identifiers can be represented by the beacon ID itself. Our post-sampling algorithm will, thus, function both for the typical fingerprinting approach as well as in a BC-to-BC approach, with the latter alternative requiring less manual effort.

In practical real-world situations, we might want to apply filtering techniques, such as a rolling window, on the last W readings from each individual AP. Algorithm 1 shows how a rolling window, which mimics a real-world situation, can be applied in a post-survey situation. In the presented algorithm, a single reference point is considered, meaning that the algorithm must be run individually for each reference point contained in the raw data. Here, we are not guaranteed to get a RSSI reading from all M APs within the sampling interval τ. Interpolation techniques, such as padding, can easily be added to fill missing readings for visible APs after this sampling algorithm has been applied. For non-visible APs with no data for the current reference point, we suggest filling missing data with φ_{min}. In our case $\varphi_{min} = -100dBm$. φ_{min} readings can work well with a ML algorithm that identifies these as certain characteristics, but should not be used directly with the distance approximation formula in Equation 2.3 or any of the mathematical localization algorithms, because the resulting distance is unlikely to match the real-world distance.

3.2.2 Capturing BC-to-BC Data

Meshtech's proprietary BLE network protocol features a functionality called beacon pinging. This functionality enables a beacon node to send advertisement packets for a period of time. The pinging interval for Meshtech beacon nodes is predefined. However, the scan-window of listening beacon nodes can be adjusted. Other listening APs in the environment will pick up these advertisement packets and compute the RSSI for each of them. The RSSI readings from all visible beacons are then routed, following the network tree hierarchy, and arrives at the GW, where a PC software can process the data. This structure is shown in Figure 3.2, where BC_p represents the pinging beacon. Advertisement packets sent by BC_p are picked

Algorithm 1 Post-survey fingerprinting data sampler

Input:

- The number of readings N and (t_i, s_i, j_i) for all $i \in \{1, 2, ..., N\}$
- The maximum number of visible APs M where $j \in 1, 2, ..., M$
- Rolling window length W
- Sampling interval τ

Output:

- List of samples S

Initiate list S
$t \leftarrow t_1$ ▷ Start time
Initiate dictionary δ with decimal values ▷ Current sample
Initiate dictionary λ with list values ▷ Sliding window
for i in $\{1, 2, ..., N\}$ **do**
    if $(t_n - t) > \tau$ **then** ▷ Timeout
        if δ contains data **then**
            $S \leftarrow S + [\delta]$ ▷ Append current sample
            Clear δ
        end if
        $t \leftarrow t_i$
    end if
    $j \leftarrow j_i$ ▷ Beacon index
    $\lambda_j \leftarrow [s_i] + \lambda_j$ ▷ Append RSSI reading
    $\lambda_j \leftarrow \lambda_{m_k}$ for all $k \in \{1, 2, ..., W\}$ ▷ Keep W previous readings
    $\delta_j \leftarrow$ average(λ_j) ▷ Apply rolling window
end for
Return S

up by all other listening and visible beacon nodes BC_j, where $j \neq p$. The purpose of the PC software is then to process and export the raw RSSI data. In a live positioning system, the PC software trained using the BC-to-BC data, can then predict the node's location on the fly.

3.2.3 Parameter Optimization

In order to adapt a localization model which performs well on the given environment, we can optimize the parameters of the radio-channel path loss propagation model presented in Chapter 2.1.3. If we assume that this model

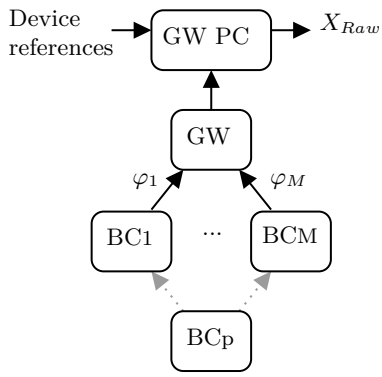


Figure 3.2: A schematic showing the flow of data during the process of capturing raw beacon-to-beacon data.

is accurate, the curvature of the path loss determined by η should depend on the environment, and thus be the same for both the BC-to-BC and the node-to-beacon scenarios. This concept can be expressed as:

$$\eta_{BC \to BC} = \eta_{ND \to BC} = \eta. \tag{3.1}$$

However, the calibrated RSSI constant, c, differ depending on the antenna characteristic, and can, thus, be different for the two scenarios:

$$c_{BC \to BC} \neq c_{ND \to BC}. \tag{3.2}$$

The calibrated RSSI in the node-to-beacon scenario $c_{ND \to BC}$ should ideally be known in advance, defined as a factory-calibrated constant, because it can be difficult to derive a good estimate for this value otherwise. If both $c_{BC \to BC}$ and $c_{ND \to BC}$ are known parameters, the path loss model can easily be adapted to both scenarios, by changing c while using the same path loss exponent η. Consequently, translating beacon-to-beacon data to a node-to-beacon scenario can thus be expressed as:

$$\varphi_{ND \to BC} \approx \varphi_{BC \to BC} - c_{BC \to BC} + c_{ND \to BC}. \tag{3.3}$$

As mentioned in Section 3.1, the BC-to-BC dataset will not include RSSI readings for distances shorter than d_{min}, where d_{min} is the distance between

the two closest beacons in the environment. Fitting the path loss model to the BC-to-BC data, by tuning both $c_{BC \to BC}$ and η, might not result in a curve that adapts well to the node-to-beacon data, because data points for shorter distances are missing, causing the curve to loose it's log-normal characteristic. In order to solve this issue, we propose locking the $c_{BC \to BC}$ to a specific value and only optimizing the η parameter. A good estimate of $c_{BC \to BC}$ should ideally be known in advance. The η can be individually tuned for each beacon, denoted as η_i, where $i \in \{1, 2, ..., M\}$. As $c_{BC \to BC}$ represents a factory-calibrated global average, it would be possible to extract an individual RSSI offset for each beacon, which we could denote as μ_i. However, this would again result in the curve potentially loosing it's log-normal characteristic. Consequently, we suggest only optimizing η in our proposed BC-to-BC approach, where good estimates of the calibrated RSSI constants, c, should be provided by the product manufacturer.

The number of RSSI readings N will not be equal for all pairs of sending- and receiving devices. Some data points, such as beacons that are close to each other can be over-represented in the raw data. Thus, simply fitting the path loss model to the raw data might not result in a model that adapts well across all distances. For this reason, we propose normalizing the number of readings for each pair of devices before optimization. This normalization can be done by instead using the mean RSSI value, $\bar{\varphi}$, obtained from each device pair.

In addition to η, an approximate of the standard deviation, σ, measured in dBm, can be obtained by analyzing the beacon-to-beacon data. When the number of measurements N goes towards infinity, e.g. $N \to \infty$, the standard deviation σ measured in dBm converges. Thus, σ can be approximated by analysing the raw data, where the pairs with the highest number of readings gives a better approximation of σ. Here, we assume that the BC-to-BC based σ can be used in the node-to-beacon scenario as well:

$$\sigma_{BC \to BC} \approx \sigma_{ND \to BC}. \tag{3.4}$$

It is also possible to derive an optimal sampling interval τ by measuring the average time interval between advertisement packets received by a single beacon. From experiments, this value seems to be close to three times the actual pinging interval of the pinging device.

The coordinates denoted in the fingerprinting dataset, can potentially be slightly inaccurate. In typical fingerprinting applications, we suggest tuning two extra parameters: a_j and b_j, which adds a linear adjustment to the calculated distances, correcting potential flaws in the reference point and beacon node coordinates. However, our experiment, indicate that performing this optimization does not translate well from the beacon-to-beacon scenario into the node-to-beacon scenario.

$$d'_j = a_j d_j + b_j. \tag{3.5}$$

When using the BC-to-BC approach, it is possible to derive parameters that adapts to the given environment, such as μ_j, σ and η_j, and possibly a_j and b_j. These parameters can be extracted automatically via an application, as long as the coordinates of the beacons χ_j are provided. The calibrated RSSIs, $c_{BC\to BC}$ and $c_{ND\to BC}$, should also ideally be known in advance, as these parameters can prove to be difficult to extract from only the beacon-to-beacon data.

3.2.4 Constructing the BC-to-BC Training Dataset

The idea with the BC-to-BC approach, is to establish a indoor localization model able to locate mobile nodes with high accuracy, only trained using RSSI values measured between beacons. As stated in Section 3.1, using only beacon-to-beacon data cause a few key challenges. One of the challenges with this approach is to transform beacon-to-beacon data so that it adapts well to the real-world node-to-beacon scenario.

We can use the proposed Algorithm 1 in order to construct a purely beacon based fingerprinting dataset, while using the logic in Equation 3.3 to adapt this dataset to a node-to-beacon scenario. However, using these data points alone to train a ML model is not sufficient, as the beacon fingerprints does not cover all possible locations of the mobile node. Thus, the ML will be good at recognizing a handful of patterns, which does not necessarily apply well in a node-to-beacon scenario. In order to overcome this issue, we propose generating extra data points which covers potential locations of the mobile node which does not exist in this fingerprinting dataset. This is where the parameters extracted from the raw BC-to-BC data, as explained

in Section 3.2.3, are finally utilized. The data points are generated by constructing a virtual environment containing both beacon and reference point positions. Thereafter, RSSI samples are generated at each virtual reference point. A simplified version of this algorithm is shown in Algorithm 2, where N samples, $X_\varphi = \{\varphi_1, \varphi_2, ..., \varphi_M\}$, are generated at each reference point, combining the parameters found in Section 3.2.3 with the propagation model in Equation 2.1.

In order to generate a realistic dataset, it is possible to further enhance the algorithm. One option would be to add the sampling algorithm shown in Algorithm 1 in order to apply a sliding window on the generated samples. However, our experiments indicated that a more robust model can be constructed by not applying any sliding window on the generated data. The generator algorithm can consider beacons out of range by defining a maximum distance d_{max} and filling the reading with φ_{min}. Additionally, the simulation can be even further extended, similar to what was done in the WILL system presented in [14], where the entire floor plan is mapped virtually. For instance, signals propagating between rooms could reduce the RSSI a certain number of dBm depending on the material of the walls. However this would significantly reduce the environment-adaptability of our approach.

As a baseline, we propose weighing the beacon fingerprints equally to the fingerprints in the generated data. This can be achieved, by joining the two types of samples into one dataset, where a factor $M/(M + R)$ of the samples are beacon fingerprints. In order to construct a unbiased model, we suggest having an equal number of samples for each reference point in all the training data.

3.2.5 Making the Final Prediction

Although different approaches can be used to make the final prediction of the mobile node's coordinates, we propose using a regression based ML algorithm, such as the RTM in order to convert the RSSI values $X_\varphi = \{\varphi_1, \varphi_2, ..., \varphi_M\}$ into distances $\hat{Y}_d = \{\hat{d}_1, \hat{d}_2, ..., \hat{d}_3\}$. The ML can hopefully see relations in the input features X_φ and derive distances that approximates the real distance, as opposed to only using the propagation model in Equation 2.1, which cannot identify said relations. Finally, a mathematical algorithm, such as least squares, can be applied to arrive at the final prediction of the node's coordinates $\chi = (x, y)$.

Algorithm 2 Simulation-based fingerprinting data generator

Input:

- The number of RPs R and coordinates χ_r for all $r \in \{1, 2, ..., R\}$
- The number of BCs M and (χ_m, η_m, μ_m) for all $m \in \{1, 2, ..., M\}$
- The desired number of samples per RP N
- Standard deviation σ
- Node-to-beacon calibrated RSSI c

Output:

- Generated samples S, distances D and target classes Y

Initialize two empty arrays S and D with shape $(N \times R, M)$
Initialize empty array Y with shape $(N \times R, 1)$
for r in $\{1, 2, ..., R\}$ **do**
    for n in $\{1, 2, ..., N\}$ **do**
        $i \leftarrow N(r-1) + n$ ▷ Index of current sample
        $Y_i \leftarrow r$ ▷ Target class
        for m in $\{1, 2, ..., M\}$ **do**
            $D_{im} \leftarrow$ Eq. 2.8(χ_r, χ_m) ▷ Distance
            $S_{im} \leftarrow$ Eq. 2.1$(c, \eta_m, D_{im}, \mu_m, \sigma)$ ▷ RSSI
        end for
    end for
end for
Return S, D and Y

Chapter 4

Improved Localization with the Tsetlin Machine

In this chapter, we present how the Tsetlin Machine (TM) can be used to enhance localization accuracy of Indoor Localization System (ILS)s. The TM can be combined with the BC-to-BC approach, presented in Chapter 3, but can also work with other approaches, such as typical fingerprinting. The concept of using ML to enhance localization in ILS is not a new concept. However, the TM is a relatively new ML algorithm, with competitive accuracy and lower computational cost compared with other ML algorithms, such as, e.g., ANNs. To our knowledge, there are currently no previous solutions in the literature that use the TM in relation to, or as a method for, indoor localization.

4.1 Binarization of RSSI Data

As stated in Section 2.4.2, the TM accepts a set of bits $X_b = \{b_1, b_2, ..., b_L\}$ as input literals. Consequently, in order to use the TM to process real values, such as RSSI readings, we must first convert the data into a binary representation. The process of converting RSSI readings into a binary representation can be expressed as:

$$X_\varphi = \{\varphi_1, \varphi_2, ..., \varphi_M\} \Rightarrow X_b = \{b_1, b_2, ..., b_L\}. \tag{4.1}$$

In this way, each of the RSSI values from sample X_φ are considered as an unique input feature, and can, thus, be converted to binary. We propose to use a typical approach for representing the RSSI in a binary format. This can be done in the following manner: Determine the minimum- and maximum possible RSSI value, then map this range linearly to B bits, where the minimum value φ_{min} is represented with only zeros (e.g., $\{0, 0, ..., 0\}$) and the maximum value φ_{max} is represented with only ones (e.g., $\{1, 1, ..., 1\}$). Any value between φ_{min} and φ_{max} can be represented by activating a certain number of bits between zero and B. The number of bits per feature, B, will affect the resolution of the input features. Assuming that the number of bits per feature B is used for all M input features, the binary representation of the sample will consequently contain $L = BM$ bits. The goal is to provide an accurate binary representation of the features in order to ensure that the model performs optimally. As stated in Chapter 2.1.2, Meshtech's devices report RSSI as an 8-bit signed integer value, where the lowest possible value $\varphi_{min} = -100dBm$. The RSSI will never exceed $0dBm$. For this reason, the raw RSSI in our case can be fully represented using $B = 100$. However, when applying filtering techniques on the raw RSSI measurements, such as a Kalman filter or a rolling window, decimal values naturally occur. In such cases using $B = -100$ may result in some degree of quantization noise.

4.2 Regression Approach

As demonstrated by Li et al. in [13], regression-based ML models can be used to improve localization accuracy in indoor localization systems. In [13], a ANN was used to convert RSSI into distances in a similar manner to what is expressed in Equation 4.3:

$$X_\varphi = \{\varphi_1, \varphi_2, ..., \varphi_M\} \Rightarrow \hat{Y}_d = \{\hat{d}_1, \hat{d}_2, ..., \hat{d}_M\}, \tag{4.2}$$

where there are M inputs and outputs, and the RSSI, φ_j is linked to the distance, $\hat{d}_j$. The idea is that the ML identifies complex relations between the input features and adjusts the distances accordingly, such that the resulting localization potentially becomes more accurate. After the RSSI has been converted into distances, the distances $\hat{d}_j$ can be used with mathematical localization approaches, such as the least squares algorithm. The least squares algorithm can consider $M \geq 3$ APs, while providing precise

localization, assuming that the predicted distances $\hat{d}_j$ are accurate. For the fingerprinting approach, the ML algorithm can recognize patterns specific for a single Reference Point (RP) and enhances the distance accuracy such that the mathematical algorithm outputs the coordinates of that reference point. Hopefully, the outputted coordinates, resembles the mobile node's real-world position.

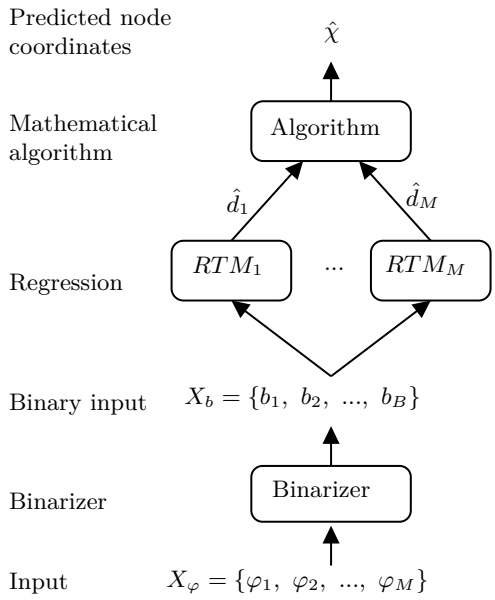


Figure 4.1: Proposed RTM-based RSSI-to-distance implementation.

As stated in Section 2.4.4, the Regression Tsetlin Machine (RTM) can be used to predict a single continuous output value $\hat{y}$ given a set of input literals X_b in a binary format. However, in order to apply a mathematical algorithm, such as the least squares algorithm, a set of distances is required. In order to obtain M distances, we propose to use M individual RTMs, where each RTM receives all input literals X_b and outputs one distance $\hat{d}_j$ each. The output of each RTM is then combined to produce the final output vector: $\hat{Y}_d = \{\hat{d}_1, \hat{d}_2, ..., \hat{d}_M\}$. Consequently, our complete model of $M\times$RTMs can still identify relations between the input features and at the same time output a set of M distances. The complete structure of the proposed implementation is shown in Figure 4.1.

In the proposed model, each RTM can be uniquely defined with individual hyper parameters, such as s, T and number of clauses m. However, for sim-

plifying construction of such a design, we suggest using the same parameters for each RTM. Thus, the total number of clauses becomes Mm. We also suggest using weighted clauses, which can significantly reduce the number of clauses required, which effectively reduces the computational costs. We can consider the set of RTMs as a single multi-output regression model. In order to train the model, the target output Y_d must be separated, such that RTM j is trained with output j. This concept is described in Algorithm 3.

Algorithm 3 Multi-output RTM training

Input:

- Binary input X_b with shape (N, L)
- Target output Y with shape (N, M)
- The number of training epochs E
- The number of clauses m
- Target T
- Precision s

Output:

- Trained RTMs Ω_j

1: $\Omega_j \leftarrow$ Initiate RTM(m, T, s) for all $j \in \{1, 2, ..., M\}$ ▷ Initiate RTMs
2: $\gamma_j \leftarrow$ Column j in Y for all $j \in \{1, 2, ..., M\}$ ▷ Split columns
3: **for** j in $\{1, 2, ..., M\}$ **do**
4: $\quad \Omega_j$.fit(X_b, γ_j, E)
5: **end for**
6: Return Ω_j

4.3 Classification Approach

An alternative to the proposed regression-based implementations, is to solve localization as a classification problem. ML based classification models can be used to predict a class i among R classes, given input vector X_φ. In this approach, each class can represent predefined locations in the environment, such as a reference point. This approach can be expressed as:

$$X_\varphi = \{\varphi_1, \varphi_2, ..., \varphi_M\} \Rightarrow \hat{y_r} \Rightarrow \hat{\chi}_r = (\hat{x}_r, \hat{y}_r). \tag{4.3}$$

The general idea behind using ML based classifier for solving indoor localization, is similar to the regression based approach; ML models can hopefully

learn and identify complex relations between the input features and more accurately predict where the target node is located.

In this alternative, we propose to use the Multi-Class Tsetlin Machine (MCTM), as it has been demonstrated to have competitive accuracy with other state-of-the-art ML algorithms. The outputted class $\hat{y}_r$ can give a rough estimate of the node's position, depending on how many predefined locations there are in the environment R, the complexity of the model, the accuracy of the training data available and potentially how the RSSI has been pre-processed. This proposed method, can be considered simpler than the regression based implementation, as we only need one MCTM with this approach. The approach can also be used with any number of APs, although more APs are usually preferred. The proposed architecture of the classification implementation is shown in Figure 4.2.

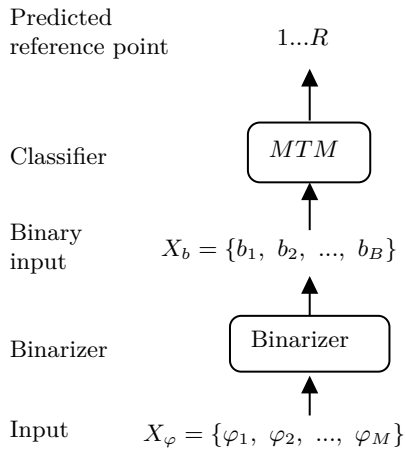


Figure 4.2: Proposed classification-based indoor localization implementation using the MCTM.

Part III

Experiments and Results

Chapter 5

Performance Evaluations

In this chapter, we present how we conducted our experiments. We will present the results obtained for the BC-to-BC approach presented in Chapter 3, and also the results obtained for our TM techniques for indoor localization that was presented in Chapter 4.

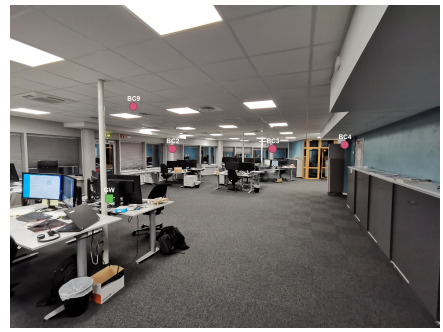
5.1 Test Environment

In order to validate the proposed methods, we sat up a real-world test bed. We constructed a Meshtech BLE network inside an office environment shown in Figure 5.1. In this experiment, we placed $M = 10$ beacon nodes at fixed locations in the environment. The beacon positions and floor map is shown in Figure 5.2. We assigned IDs to the beacons, e.g., BC_0, and measured their coordinates in three Cartesian dimensions with centimeter precision. In order to minimize congestion in the network, all of the beacons were connected directly to the GW. In other words, all beacons were direct child nodes of the GW. We temporarily set the beacon devices' scan window to the maximum, specifically 95%, to capture as many advertisement packets as possible during the data capturing process. In comparison, the default scan window for these devices is 30%.

In order to capture and export the raw RSSI measurements, we developed a unique software that exports the raw RSSI measurements. This software

handles communication with the GW device through a serial port. In addition, we made a helpful script to streamline the workflow of assigning IDs and reference points with this software. Figure 5.3 is a schematic representation of our setup. For the beacon-to-beacon data, the script iterated through each beacon node and enabled pinging from that beacon node for a period of time. In addition to the 10 beacon nodes, 4 mobile nodes were used to capture fingerprinting data among $R = 87$ reference points. We placed the mobile nodes on stands approximately $1.34m$ above the floor for all reference points. The reference point positions are also denoted in the floor plan in Figure 5.2. We set the advertisement interval for the nodes to $200ms$ during the data capture. The process of capturing beacon-to-beacon data was fully automated using the script, while the node-to-beacon data required re-positioning of the mobile nodes during the process. The reader can find the complete list of devices used in our test bed in Appendix A.

(a) Left



(b) Right

Figure 5.1: Photos of our test environment.

5.1.1 Analysis on the Raw Data

At each reference point, we captured data for five minutes. The entire capturing process took about nine hours. At the same time, we continuously captured beacon-to-beacon data. Each beacon node was pinging for 30 seconds each, one at a time. The resulting dimensions of the captured raw RSSI measurements can be found in Table 5.1.

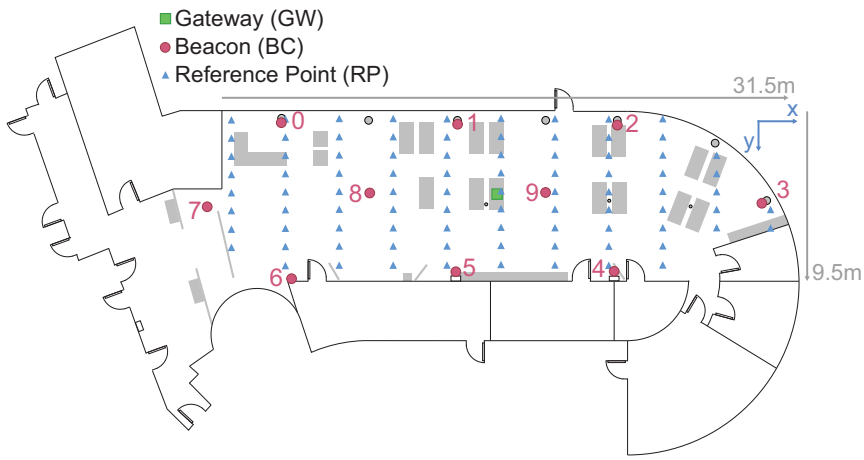


Figure 5.2: Floor plan of our test environment.

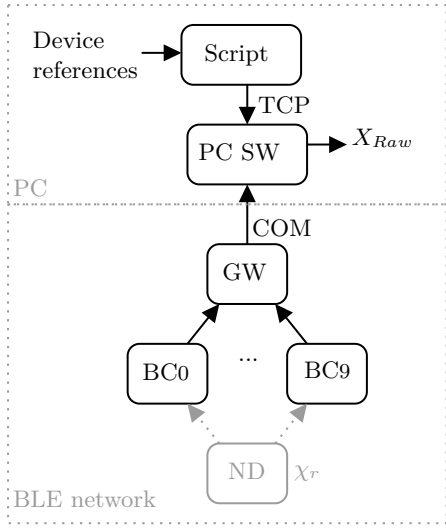
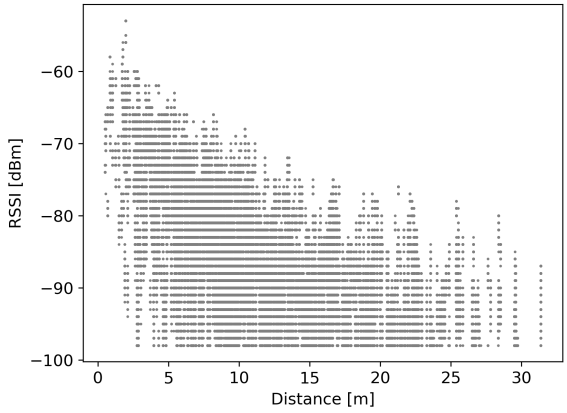


Figure 5.3: A schematic showing the flow of data during the raw data capturing process.

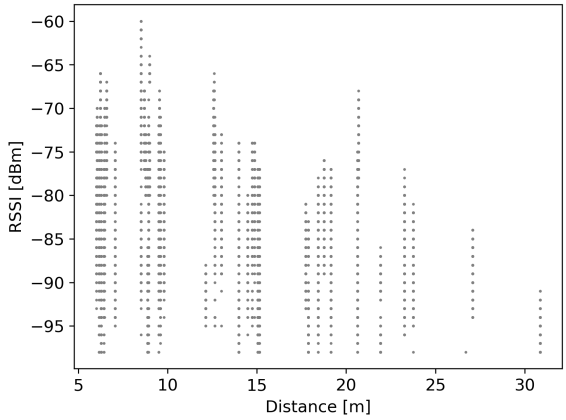
The raw RSSI measurements are plotted in Figure 5.4 for both the node-to-beacon and beacon-to-beacon scenarios, respectively. In the two Figures, the x-axis is the true distance d in meters, while the y-axis is the RSSI measured. Noise was highly prominent in the raw data, as we can observe from the figure. For the beacon-to-beacon data, the log-normal characteristic was not very apparent.

Table 5.1: The dimensions of our raw data.

Direction	Nr. of Measurements
Node-to-Beacon	432,017
Beacon-to-Beacon	26,160
Total	458,177



(a) Node-to-beacon

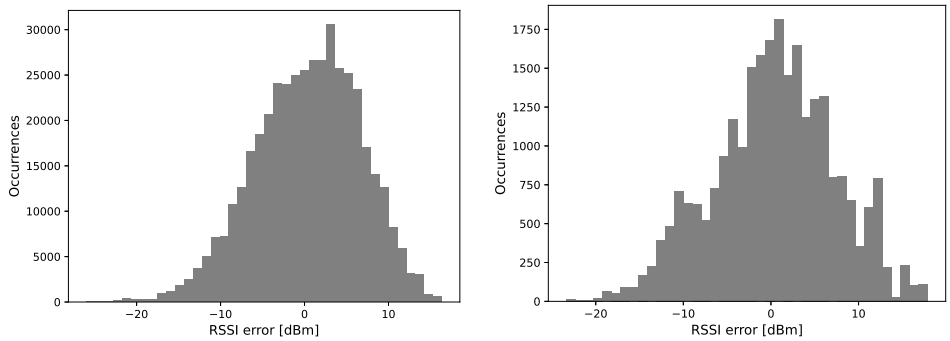


(b) Beacon-to-beacon

Figure 5.4: The figures visualize each individual data points in our raw data.

Figure 5.5 displays the RSSI error distributions after tuning the parameters of the propagation model presented in Section 2.1.3. The error is calculated

as $(\varphi_{Measured} - \varphi_{Expected})$, where $\varphi_{Expected}$ is the expected RSSI given the real distance d, according to the propagation model. Both of the histograms follow a normal distribution, which is expected, according to the theory of radio channel propagation characteristics presented in Section 2.1.3, which states that the noise A_σ is a Gaussian distributed random variable.



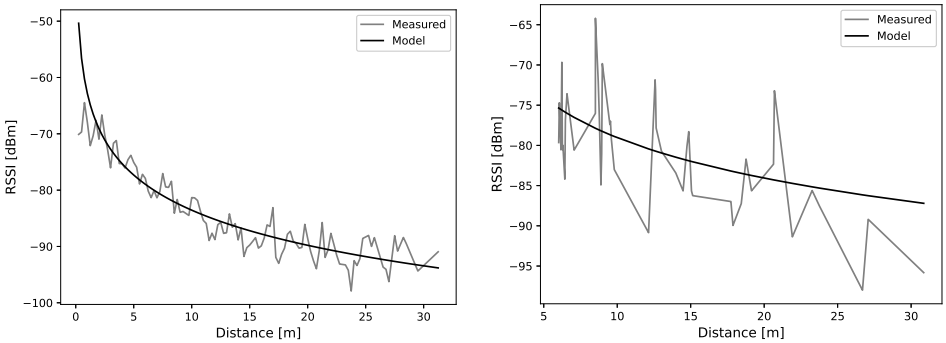
(a) Node-to-beacon (b) Beacon-to-beacon

Figure 5.5: RSSI error distribution in the raw data. Here, the error is calculated by comparing the measured RSSI with the expected RSSI. The expected RSSI is calculated using the radio-channel path loss model (Equation 2.2) with globally optimized parameters (c and η), provided the distance, which we know.

5.1.2 Parameter Optimization

In this section, the parameters of the propagation model presented in Section 2.1.3 are optimized based on the raw data. Please note that the optimized parameters presented in this section could have been slightly different if the analysis was performed on pre-sampled data, especially if the sampled dataset had a significant amount of data interpolation. Figure 5.6 illustrates how the propagation model in Equation 2.1 can be adapted to fit the data by tuning its parameters c and η. Table 5.3 shows the parameters used to plot the model. We expected to find a similar η in both scenarios because the environment is the same. However, this was not true, because in this case, $\eta_{BC \to BC} < \eta_{BC \to BC}$. The $\eta_{BC \to BC}$ was even lower than what was expected for a free-space environment. The beacon-to-beacon data did not include readings for distances closer than $d_{min} = 6.02m$, which could have impacted the curve fitting process. In order to mitigate the issue with

mismatching η, we determined that the best solution was to set $c_{BC \to BC}$ to a statically defined value. The reader should note that the calibrated RSSI in neither of the directions was known in advance to our experiments. We assumed that the parameters in the node-to-beacon scenario were ideal. Because the data was readily available, we used the $\eta_{ND \to BC}$ in order to find an ideal $c_{BC \to BC}$. By doing this, we arrived at $c_{BC \to BC} = -58.54dBm$ which we believed to be much closer to the actual value.



(a) Node-to-beacon (b) Beacon-to-beacon

Figure 5.6: The signal path loss model fitted on our raw data. In these figures, the log-normal characteristic of the raw data was highlighted by plotting the average measured RSSI given the distance between the devices.

Table 5.2: Radio-channel path loss model parameters (c and η) optimized for the node-to-beacon and beacon-to-beacon scenarios, respectively. The parameters were optimized using two Cartesian dimensions. The values in the bottom row were found by considering $\eta = 2.15$ found in the node-to-beacon scenario as the ground truth and then exclusively optimizing c.

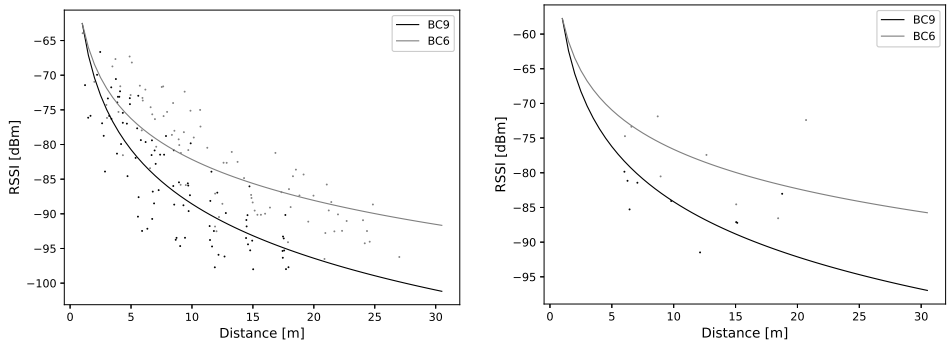
Direction	c	η
Node-to-beacon	$-63.33dBm$	2.07
Beacon-to-beacon	$-60.74dBm$	1.87
Beacon-to-beacon'	$-58.54dBm$	2.07

Table 5.3 shows the resulting η_j individually optimized per beacon, with c globally set. The data in this table suggest that the path loss exponent η is fairly similar in both the node-to-beacon and beacon-to-beacon scenario across all individual beacons. In respect to the BC-to-BC approach,

these results are promising, because they suggest that the η retrieved from a beacon-to-beacon scenario can be directly applied in a node-to-beacon scenario. Figure 5.7 shows the propagation model plotted for two individual beacons using the parameters presented in Table 5.3, which illustrates that the curve show some resemblance between the two scenarios.

Table 5.3: Radio-channel path loss model parameters (η and σ) optimized for each beacon, individually.

Beacon #	Node-to-beacon		Beacon-to-beacon	
	η	σ	η	σ
All (0–9)	2.07	3.38	2.07	4.06
0	1.98	3.48	1.92	4.29
1	2.21	3.77	2.07	4.33
2	2.11	3.64	1.85	3.33
3	2.45	2.22	2.44	2.98
4	2.11	3.56	1.92	4.48
5	1.90	3.45	2.10	5.23
6	1.89	3.00	1.81	3.68
7	1.79	2.94	2.02	3.83
8	1.89	3.26	1.98	3.86
9	2.53	3.33	2.57	4.45



(a) Node-to-beacon (b) Beacon-to-beacon

Figure 5.7: The radio-channel path loss model individually fitted per beacon. The figures depicts data from two beacons. The dots represent the mean RSSI value measured between a advertising device, located at a specific coordinate, and the beacon.

The standard deviation, σ, for the beacon-to-beacon scenario is on average $0.78dBm$ higher than for the node-to-beacon data. As stated in Section 3.2.3, σ should converge to the correct value when the number of readings goes toward infinity. From this, the σ can be approximated. However, it was concluded that the average standard deviation from all beacon-to-beacon measurements, namely $4.06dBm$, was sufficient for our node fingerprinting data generator.

5.2 Data Preparation

5.2.1 Beacon Fingerprints

We constructed a pure beacon-to-beacon based fingerprinting dataset by sampling the raw beacon-to-beacon signal with our proposed post-survey sampling algorithm presented in Section 3.2.1. We measured the average measurement interval to be $2.14s$, thus we used $\tau = 2.14s$ as our sampling interval in order to minimize data loss. After the sampling process, we normalized the number of samples per beacon, so that each beacon beacon's reference point weighed the same in the training data. Each RSSI reading was adapted to a node-to-beacon scenario using Equation 3.3. One of the pairs, namely BC_3 and BC_6, never picked up any RSSI readings from each other. The missing pair, approximately 1.1% of the dataset, was filled using $\varphi_{min} = -100dbm$. After this, the remaining empty values in the data was filled with a padding interpolation technique. We refer to the resulting beacon fingerprinting data as $\mathbf{Train}_{BC}$.

5.2.2 Node Fingerprints

We used the node-to-beacon data in order to construct testing data for our BC-to-BC approach. We refer to this validation data as $\mathbf{Test}_A$. We reserved five randomly selected reference points for a separate testing dataset, referred to as $\mathbf{Test}_B$. Consequently, Test_A, contains samples from 82 reference points. This testing data gives a good indication of the model's real-world performance. The same 82 reference points were used to construct training- and validation data for our regular fingerprinting models, referred to as $\mathbf{Train}_{FP}$ and $\mathbf{Test}_{A'}$, respectively. We ensured strict separa-

tion between training and validation data. This was achieved by separating the raw node-to-beacon data in the time domain before applying our sampling algorithm. Thus, none of the samples in Train_{FP} and $\text{Test}_{A'}$ are based on the same raw data. Test_B consist of reference points which are unfamiliar to the models, and can be used to detect potential model *overfitting*[1]. We argue that by reserving some of the reference points for testing, we can analyze the model's performance on unseen coordinates, or coordinates that are located somewhere between the reference points, which are not present during training. For all the node-to-beacon data, we tried to construct authentic samples that closely resemble data similar to what we would observe from a real-world application. Here, we applied a rolling window with length $W = 6$. We also filled non-visible beacons with $\varphi_{min} = -100dBm$. In addition, we normalized the number of samples per pair so that they had the same weight during testing.

5.2.3 Simulated Datapoints

We built the simulation based training dataset using the dataset generation algorithm presented in Section 3.2.4. Consequently, we simulated RSSI readings by using the parameters presented in Table 5.3, which we obtained through the BC-to-BC approach. Any coordinate within the room boundaries could in fact be simulated. Here, we simulated the same 82 real-world reference points and generated $N = 300$ samples for each of them. We did not apply any rolling window or other filtering techniques on this generated data. This was done in order to reduce the models' over-confidence on the training data, as we could not be certain that the simulated data accurately described every reference point in the real-world environment. The resulting dimensions for our sampled datasets are stated in Table 5.4.

[1]If a model is overfitted, it means that it is specially adapted to the training data and that it does not work in a generalizing way. Such a model will perform very poorly on new cases. Appropriate testing of the model can help us detect overfitting, e.g., by testing its performance on data from devices entirely omitted in training. We would expect very good results in training and validation for an overfitted model but very poor results on separate test data.

Table 5.4: The dimensions of our sampled training- and validation data.

Approach	Dataset name	RPs	Samples	Data source
BC-to-BC	Train_{BC}	10	2,580	Beacon fingeprints
	Train_{PL}	82	26,100	Simulation
	Test_{A}	82	33,620	Node fingerprints
Any	Test_{B}	5	2,000	Node fingerprints
FP	Train_{FP}	82	29,848	Node fingerprints
	$\text{Test}_{A'}$	82	2,952	Node fingerprints

5.3 Result for the TM

In this section, we present the experiments and results of the proposed TM based localization techniques presented in Chapter 4. For our RTM approach, a total of $M = 10$ Weighted RTMs (W-RTM) were used, where each W-RTM corresponded to one of the beacons in our test bed.

Table 5.5: Selected hyper parameters for our TM implementations. The abbreviation W refers to the use of weighted clauses.

Model	m	T	s	Bits per feature	Epochs
W-RTM ($\times 10$)	360	1800	2.0	200	30
W-MCTM	200	500	1.5	500	10

5.3.1 Hyper Parameter Search

The performance and complexity of the TM depends on the hyper parameters used, such as the target value T, the precision s, the number of clauses m, and the granularity of the input features, here determined by the number of bits per feature B. We conducted experiments in order to derive optimal parameters for the RTM based solution. In our case, we set T to be five times m, e.g., $T = 5m$. In order to derive an appropriate s, we conducted a hyper parameter search, where different values for s and m were mapped in a so-called *heat map*. The resulting hyper parameter search heat map is shown in Figure 5.8. The hyper parameters we ended up using are shown in Table 5.5.

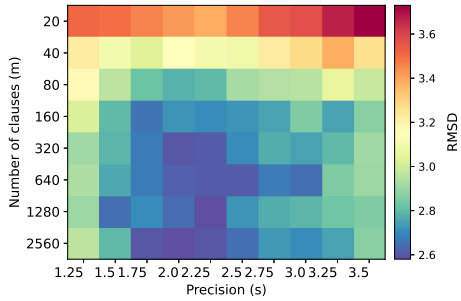
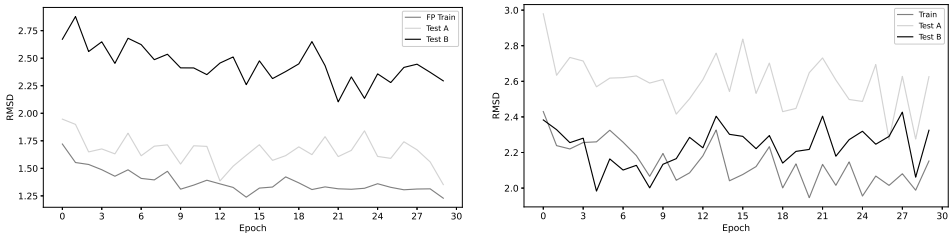


Figure 5.8: Hyper parameter search heat map for our RTM based RSSI-to-distance model. Here, the RMSD is evaluated using the Test_A dataset. The model was trained for 10 epochs using BC-to-BC training data, with 80% samples from Train_{PL} (e.g., 80% simulated training data).

5.3.2 Model Training

Figure 5.9 demonstrates the training history for our 10×W-RTM based RSSI-to-distance model over 30 training epochs. These figures are based on a single run, and visualize how the training data differ from the test data.



(a) Fingerprinting (b) BC-to-BC

Figure 5.9: Training history for our W-RTM RSSI-to-distance implementation.

5.3.3 ANN Implementation

We chose to compare the performance of our RTM implementation with an ANN. In order to make our numerical results repeatable we describe our ANN implementation in this section. Table 5.6 show the parameters used

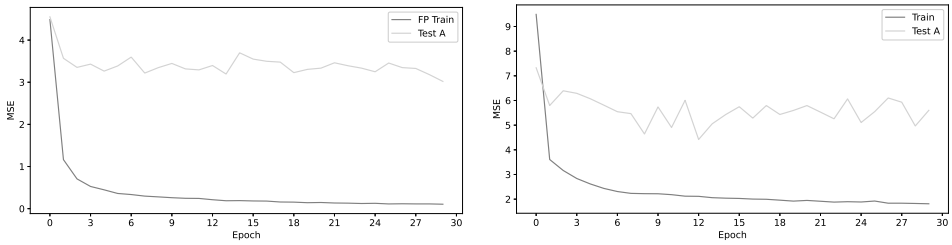
for our ANN implementation, while Table 5.7 describes the layer structure. In total, the regression implementation had 495,520 trainable parameters. The model had $M = 10$ inputs and $M = 10$ outputs. This way, the ANN cpuld convert each sample $X_\varphi = \{\varphi_1, \varphi_2, ..., \varphi_M\}$ into M distances $\hat{Y}_d = \{\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_M\}$. Figure 5.10 shows the training history for our ANN based RSSI-to-distance model. We used a similar layer structure for our ANN based classification model. The only difference was that we used a soft-max activation on the output layer and a categorical cross-entropy optimizer. The classification model had $R = 82$ outputs, corresponding to the 82 real-world reference points in Test_A.

Table 5.6: Parameters used for our ANN implementations.

Type	Optimizer	Epochs	Batch size
Regression	Adam	30	30
Classification	Categorical cross-entropy	30	30

Table 5.7: The layer structure of our ANN implementations. Layer 4a represents the output layer of our ANN regression model, while 4b represent the output layer of our ANN classification model.

Layer	Activation	Neurons	Params
1. Dense	ReLU	10	110
2. Dense	ReLU	800	8,800
3. Dense	ReLU	600	480,600
4a. Dense	Linear	10	6,010
4b. Dense	Soft-max	82	49,282



(a) Fingerprinting (b) BC-to-BC

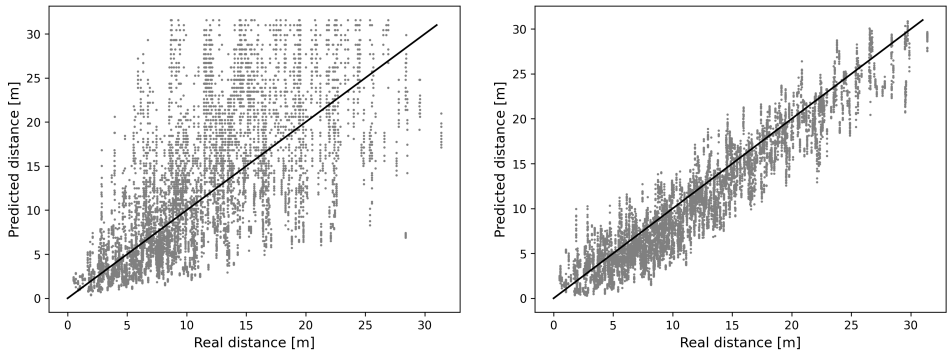
Figure 5.10: Training history for our ANN regression model.

5.3.4 Resulting Performances

In Table 5.8 we present the resulting performance for our regression based ML models. The resulting performances are measured using RMSD, where lower values indicate lower overall error and better performance. The RMSD relates to the resulting localization accuracy. However, this relation may not be completely linear, depending on which mathematical algorithm is applied to derive the final prediction of the target node's location. Figure 5.11 plots the RSSI-to-distance conversion made by the standard path loss model versus our W-RTM model's predictions, on the same data. This figure illustrates how ML can be used to enhance our distance estimates.

Table 5.8: Resulting RMSD for our trained regression models.

Model	FP		BC-to-BC	
	$\text{Test}_{A'}$	Test_B	Test_A	Test_B
W-RTM	1.58m	2.26m	2.52m	2.52m
ANN	1.64m	1.99m	2.30m	2.25m



(a) Path loss model (Equation 2.3) (b) W-RTM

Figure 5.11: RSSI-to-distance conversion on the Test_A data. The black line represents a perfect prediction. Each dot represent the result of a single RSSI-to-distance conversion. (a) uses the standard path loss model without AI. (b) uses our trained W-RTM machine learning model.

5.4 Resulting Localization Accuracy

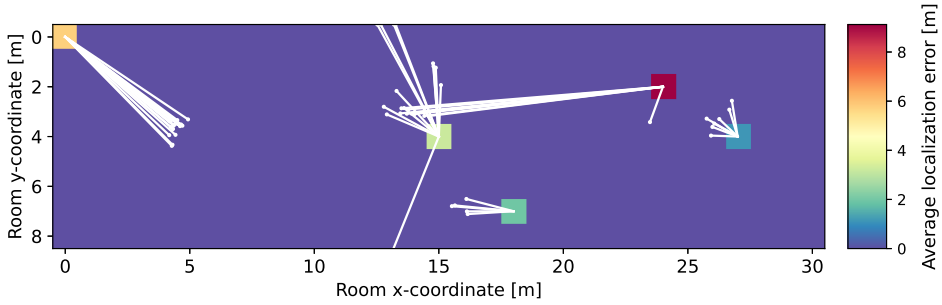
Because our system is non-deterministic and partly rely on random chance, the numerical results presented in this section are based on the average from multiple, in total ten, separate runs, using consistent parameters. For each experiment, five different reference points were randomly selected to be excluded from the training data, i.e., we re-compiled the datasets and retrained the models in each run.

Table 5.9: Resulting localization accuracy for trilateration, least squares and WCL without machine learning. The *No optimization* column serves as a general reference, using calibrated RSSI $c = -50dBm$ and path loss exponent $\eta = 3$.

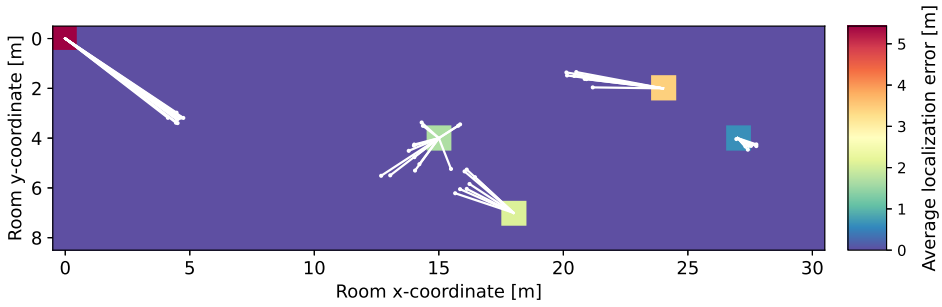
Alg.	FP		BC-to-BC		No optimization	
	$\text{Test}_{A'}$	Test_B	Test_A	Test_B	Test_A	Test_B
Tri.	4.95m	4.60m	4.85m	4.67m	4.94m	5.19m
LS	4.91m	4.99m	5.36m	5.27m	6.09m	7.03m
WCL	2.52m	2.68m	2.64m	2.82m	2.85m	2.85m

5.4.1 Without Machine Learning

In Table 5.9, we present the resulting localization accuracy for three typical mathematical localization algorithms, namely trilateration, LS and WCL. The RSSI data was converted to distances using Equation 2.3. We tested each of these algorithm using parameters found trough the fingerprinting approach and our own BC-to-BC approach. The path loss exponent η was individually adjusted for each beacon based on the optimized parameters shown in Table 5.3. For the WCL algorithm, we used attraction field factor $g = 2$. The reader should note that the implementations of the traditional mathematical algorithms are implemented in a manner which works in favour of these algorithms. For instance, the prediction for these models are ensured to stay within the boundaries of the room. We ensured that the filling RSSI of $\varphi_{min} = -100$ [dBm] was never used with any of the algorithms. Thus, we argue that the numeric results presented in the table should be considered to be near the best achievable localization accuracy for these algorithms. Figure 5.12 illustrate predictions made by the trilateration and WCL algorithm on Test_B without using ML.



(a) Trilateration



(b) WCL

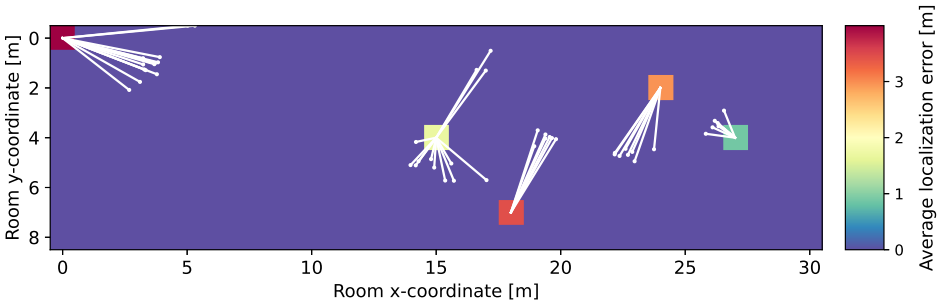
Figure 5.12: Visualization of predictions made on the Test_B data using trilateration and WCL, respectively. In these experiments, we used the path loss parameters (c and η) individually tuned for each beacon obtained with the BC-to-BC approach. The colored cells represent the average localization error, in meters, for a single reference point. The white dots represents 50 predictions made by the model. The white lines indicate the distance between those predictions and the target node's true location.

5.4.2 With Machine Learning

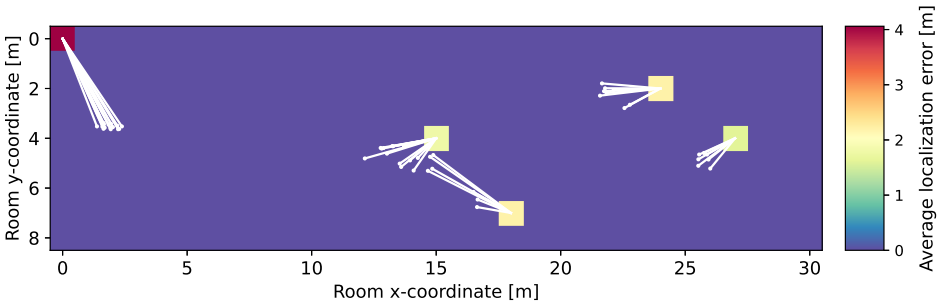
In Table 5.10, we present the resulting localization accuracy for our ML-based implementations. The three mathematical algorithms, trilateration, least squares and WCL were used to determine the node's coordinates. Figure 5.13 visualizes 50 predictions done by our W-RTM based RSSI-to-distance model on the Test_B dataset.

Table 5.10: Resulting localization accuracy for our machine learning-based implementations. Here, the best result for each column is highlighted.

ML	Alg.	FP		BC-to-BC	
		$\text{Test}_{A'}$	Test_{B}	Test_{A}	Test_{B}
W-RTM	Tri.	1.51m	2.68m	3.00m	3.36m
	LS	1.53m	**2.67m**	3.00m	**2.72m**
	WCL	1.84m	3.05m	3.24m	3.08m
ANN	Tri.	**0.63m**	2.94m	2.80m	2.91m
	LS	0.69m	2.89m	**2.79m**	2.82m
	WCL	1.84m	3.05m	3.24m	3.08m



(a) Fingerprinting



(b) Beacon-to-beacon

Figure 5.13: Visualization of predictions made on the Test_{B} data using our RTM-based implementation applying the trilateration algorithm.

Table 5.11: Resulting classification accuracy for our classification based localization models.

Model	FP	BC-to-BC
	$\text{Test}_{A'}$	Test_{A}
W-MCTM	100.0%	14.8%
Naive Bayes	88.6%	16.8%
Random Forest	99.0%	15.0%
ANN	94.4%	13.4%

5.5 Resulting Classification Accuracy

The results on the classification models are shown in Table 5.11. Here the MCTM is compared with the naive Bayes classifier, random forest classifier and an ANN. For our BC-to-BC approach, we trained our models using Train_{PL} data with generated data for the same 82 reference points as Test_{A}. Test_{B} contains none of the same reference points and are therefore not used for evaluation of these models. When measuring the performance of classification models, the percentage of correct guesses among N samples, is commonly used as a validation metric. Thus, classification models become easy to compare provided the same dataset. With fingerprinting or similar approaches, it is however possible to use the reference point's true position to calculate the model's average localization error. However, as each correct guess will result in $0m$ localization error, the average localization error will potentially become unrealistically low. For this reason, we argue that the average localization error of classification-based models cannot be directly compared with the localization error of regression based- and mathematical approaches. Consequently, only the resulting percentages are presented in this table.

5.6 Discussion

The results presented in this chapter, allows us to compare the typical fingerprinting approach with our proposed BC-to-BC approach. We argue that the numerical results presented in this chapter are, to our best efforts, unbiased. For instance, the models used for both the fingerprinting- and the BC-to-BC approach are the same. The only difference is the training

data, where the BC-to-BC models were trained using beacon fingerprints and simulation data, and the fingerprinting models were trained using real-world node fingerprints. For the evaluation data, each reference point had an equal number of test samples. Consequently each reference point weigh the same in the results. We ran the experiments multiple times, and the values presented are based on the average result from ten experiments.

The results on the Test_B data gave us an indication of the models' performance on locations in-between the reference points of the training data (thus, *unknown* points). Here, our RTM-based RSSI-to-distance model achieved an average localization accuracy of $2.67m$ using fingerprinting and $2.72m$ ($0.05m$ worse) using BC-to-BC. At the same time, our ANN-based fingerprinting model achieved $0.63m$ average localization error on the $\text{Test}_{A'}$ dataset, which utilizes the same 82 reference points as in the training data. This is far superior to that of the BC-to-BC approach, which achieves $2.79m$ average localization error on the same data. Consequently, we argue that the localization accuracy achievable by the fingerprinting approach lays somewhere between $0.6m$ and $2.7m$, while for the BC-to-BC approach, we are more confident that the accuracy is near $2.8m$, at least on our data. The results on the Test_B testing dataset indicates that the fingerprinting approach does not achieve the expected localization accuracy for any coordinate that was not included in the training data. Because of this, as well as the intensive manual labor required to construct fingerprinting training data, we believe the BC-to-BC approach is promising, although the achieved $2.8m$ average localization error is not very satisfying. We believe that the localization accuracy can be further enhanced through future work, and that our experiments serve as evidence that re-calibrating ILSs using beacon-to-beacon data is feasible.

The BC-to-BC approach is still arguably more adaptive, because it does not require re-positioning of the nodes during the offline phase. We believe that methods from the proposed BC-to-BC approach can be used in applications where the typical fingerprinting is not applicable and where the localization accuracy requirement is slightly lower. One apparent weakness with the BC-to-BC approach, is that good estimates for the calibrated RSSIs, $c_{BC \to BC}$ and $c_{ND \to BC}$, should be known in advance, and that these are difficult to acquire trough the beacon-to-beacon data alone. As a consequence, the product manufacturer would likely need to conduct lab experiments in order to provide an accurate calibrated RSSI for their products. Hopefully, the calibrated RSSIs found through our experiments, can be applied in future

real-world Meshtech ILSs for their specific devices, or for devices with similar antenna characteristics.

In Table 5.11, we present the results on the classification models. Here, our MCTM seem to achieve slightly better accuracy compared with the other algorithms using the fingerprinting approach. However, using our simulated Train_{PL} dataset, which features the same 82 reference points as Train_{FP}, all models appear to struggle with identifying the correct reference point. This indicates that our generated data does not match it's real-world counterpart to a sufficient degree in order to achieve the same level of accuracy as fingerprinting. Still, by lowering the number of classes, enlarging each cell, the BC-to-BC approach can still function to some degree.

Using the regression ML models, such as the RTM, to process the RSSI can drastically improve the localization accuracy of the mathematical algorithms, such as trilateration and least squares, as is demonstrated through our experiments. However, we feel that it is important to clarify that single-RSSI-to-distance conversion can be achieved with simple mathematical expressions, such as the typical propagation model presented in Equation 2.2. ML provides improved localization by recognizing relations between RSSI received by all beacons and can then adjust it's predictions accordingly. The reader should note that the WCL algorithm achieved great results on our test data, compared with the other implementations. In fact, WCL using BC-to-BC parameters achieves $0.15m$ more accurate localization than our BC-to-BC-based ANN implementation on the Test_A dataset. For this reason, we recognize WCL as a strong competitor to the ML based solutions. The reader should note that the WCL algorithm function well where beacons surrounds the environment, e.g., in every corner of a square room, and where the point of origin is set to the center coordinate between all APs (which is the case for our test environment). Our experiments indicate that the WCL is a valid option in scenarios where the beacons are placed in a similar manner to our test bed. Although not validated through our experiments, we hypothesize that in environments where the beacons do not captivate the room completely, ML can be used to add additional beacons, virtually, similar to how we simulate reference points in our BC-to-BC approach. This would enable the mathematical algorithms, such as WCL, to predict coordinates outside their typical boundaries.

In our experiments, we tried to implement both the RTM and the ANN with a similar level of complexity. According to our results presented in Table 5.10, our RTM based implementation achieves similar test performance

as our ANN implementation[2]. However, in regards to the $\text{Test}_{A'}$ data, the ANN show better performance, as it achieves $0.6m$ average localization error, in contrast to 1.5 achieved by the RTM. However, we believe that with more hyper tuning, we might have achieved even better results than what we presented earlier. The reader should note that the TM is known to be less resource intensive than the ANN, which has also been numerically shown by Lei, Et al. in [34]. In addition, TMs can hypothetically be constructed using electric circuits consisting of basic logic gates. In fact, such solutions have already been prototyped using FGPA by Wheeldon et. al, as explained in [5]. Therefore, the TM algorithms can potentially be placed as an on-chip component, doing ML with near instantaneous prediction and low power consumption. For this reason, using TM to pre-process RSSI directly on-chip, is plausible and an exiting concept.

[2]The number of bits required to represent our RTM model is $M \times 2Lm = 2BM^2m = 14.40Mb$, while the ANN regression implementation requires approximately $595520 \times 32 \approx 15.86Mb$, as each weight can be represented by a 32 bit floating point decimal value.

Chapter 6

Conclusion and Future Work

This chapter summarizes and concludes the work in this thesis. In Section 6.2 we also discuss how the proposed methods can be further enhanced.

6.1 Conclusion

Although GPS provide superior localization accuracy outdoors, GPS systems have major difficulties providing sufficient localization accuracy in indoor settings. Indoor localization using wireless signals has become an interesting area of research and is still highly ongoing. In this thesis, we presented a novel strategy for extracting beacon-to-beacon RSSI readings and we proposed methods for utilizing this data as a foundation to build and train fingerprinting based localization models. While, the fingerprinting method requires extensive manual labor during the offline phase, the proposed BC-to-BC method allows the the Indoor Localization System (ILS) to be deployed directly from the application with much less involvement from end users and system administrators. The only requirement is to have a good estimate of the devices' factory calibrated RSSIs and AP coordinates. Based on the experiments done in this study, we conclude that by using the proposed BC-to-BC approach combined with machine learning, we can achieve better localization accuracy, approximately $2.8m$, compared with orthodox localization algorithms, such as trilateration or least squares, which achieves roughly $4.9m$ in comparison. The proposed methods did

not outperform the well-established fingerprinting method, which achieves somewhere between $0.6 - 2.7m$ [1] localization accuracy on the same data. Still, the results are promising because the proposed method by nature is arguably more adaptive than fingerprinting. With further hyper parameter tuning of the TM, the proposed methods might achieve even better results than the ones presented in this thesis. The TM is less computationally complex and less memory hungry than its main competitor, the ANN. Therefore, the proposed TM based methods for indoor localization constitute interesting competitors within this area of research. We also consider WCL a solid competitor to the ML-bases approaches, because it achieves $2.7m$ average localization accuracy on our test data when optimized with the BC-to-BC approach.

6.2 Future Work and Enhancements

The future enhancements for the BC-to-BC approach involve adapting our methods so that they can be used in real-time localization systems. In this thesis, we apply our BC-to-BC approach in an environment where the nodes are statically positioned in the environment. For live localization systems, however, more advanced filtering techniques, such as the Kalman filter, can be applied to further enhance the localization accuracy of moving objects. Applying recurrent ML models, such as RNNs or the anticipated Recurrent TM, is considered to be the next big step in enabling the BC-to-BC approach to handle time series data. In addition, the newly introduced Bluetooth directional finding can also be combined with our approach to further enhance the localization accuracy [2]. In further testing of the BC-to-BC approach, we plan to evaluate its re-calibration capabilities and adaptability in environments that change over time. For instance, we could physically modify the environment, e.g., by moving physical objects, thereby blocking and altering some of the signals in the environment to further test the adaptability of the system.

[1] While the fingerprinting approach achieves $0.6m$ localization accuracy on known reference points, it achieves approximately $2.7m$ localization accuracy at best on coordinates not included in the training data. For this reason, we represents its real-world performance as a range.

[2] The reader should note that Bluetooth directional antennas have just recently become available on the market.

The WCL algorithm achieved impressive localization accuracy in our test environment. However, the WCL algorithm might not achieve similar performance in environments with different beacon configurations. We hypothesize that ML algorithms, such as the RTM, could be used to add additional virtual beacons, thus allowing the mathematical localization algorithms to predict locations outside their typical boundaries. This concept could potentially extend the capabilities of our BC-to-BC approach, and should be further investigated.

Chintalapudi et al. [6] has suggested dynamically determining the location of a mobile node through other measures, such as GPS or NFC signals at known sites. In this way, FP data can be obtained without any pre-survey, which is a strategy that would be interesting to explore in further work. In this context, another exciting enhancement is implementing a similar process to automatically retrieve more accurate FP training data directly from the environment.

The BC-to-BC approach currently requires the beacon coordinates to be known in advance. However, Wu et al. [14] suggests that we can approximate the locations of all APs using their approach without any on-site survey. This approach does, however, rely on the complete virtualization of the building's floor plan. We believe that the beacon coordinates can be estimated through the means of beacon pinging for the BC-to-BC approach. However, the complexity of this task, especially for multi-floor and multi-room environments, makes the job very challenging and time-consuming. Therefore, we did not consider this aspect in this thesis, but we should investigate it in further work.

References

[1] M. G. Wing, A. Eklund, and L. D. Kellogg, "Consumer-Grade Global Positioning System (GPS) Accuracy and Reliability," *Journal of Forestry*, vol. 103, pp. 169–173, June 2005.

[2] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, "Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 1327–1346, 2017. Conference Name: IEEE Communications Surveys Tutorials.

[3] O.-C. Granmo, "The Tsetlin Machine – A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic," *arXiv:1804.01508 [cs]*, Jan. 2021. arXiv: 1804.01508.

[4] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, pp. 431–440, July 2015.

[5] A. Wheeldon, R. Shafik, T. Rahman, J. Lei, A. Yakovlev, and O.-C. Granmo, "Learning automata based energy-efficient AI hardware design for IoT applications," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, p. 20190593, Oct. 2020. Publisher: Royal Society.

[6] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, MobiCom '10, (New York, NY, USA), pp. 173–184, Association for Computing Machinery, Sept. 2010.

[7] N. A. K. Zghair, M. S. Croock, and A. A. R. Taresh, "Indoor Localization System Using Wi-Fi Technology," *Iraqi Jorunal of Computers, Communications, Control & Systems Engineering (iJCCCE)*, vol. 19(2), pp. 69–77, Feb. 2019.

[8] Z. Ma, S. Poslad, J. Bigham, X. Zhang, and L. Men, "A BLE RSSI ranking based indoor positioning system for generic smartphones," pp. 1–8, Apr. 2017.

[9] S. He and S.-H. G. Chan, "Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 466–490, 2016. Conference Name: IEEE Communications Surveys Tutorials.

[10] O. Oguejiofor, A. Aniedu, E. H.C, and O. A.U, "Tilateration Based Localization Algorithm for Wireless Sensor Network," *IJISME*, vol. 1, Sept. 2013.

[11] M. Woolley, "Bluetooth Direction Finding - A Technical Overview," Feb. 2021.

[12] F. Zafari, A. Gkelias, and K. K. Leung, "A Survey of Indoor Localization Systems and Technologies," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019. Conference Name: IEEE Communications Surveys Tutorials.

[13] G. Li, E. Geng, Z. Ye, Y. Xu, J. Lin, and Y. Pang, "Indoor Positioning Algorithm Based on the Improved RSSI Distance Model," *Sensors (Basel, Switzerland)*, vol. 18, Aug. 2018.

[14] C. Wu, Z. Yang, Y. Liu, and W. Xi, "WILL: Wireless Indoor Localization without Site Survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 839–848, Apr. 2013. Conference Name: IEEE Transactions on Parallel and Distributed Systems.

[15] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: unsupervised indoor localization," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, (New York, NY, USA), pp. 197–210, Association for Computing Machinery, June 2012.

[16] T. Lovett, M. Briers, M. Charalambides, R. Jersakova, J. Lomax, and C. Holmes, "Inferring proximity from Bluetooth Low Energy RSSI with

Unscented Kalman Smoothers," *arXiv:2007.05057 [cs, eess, stat]*, July 2020. arXiv: 2007.05057.

[17] R. Faragher and R. Harle, "Location Fingerprinting With Bluetooth Low Energy Beacons," *IEEE Journal on Selected Areas in Communications*, vol. 33, pp. 2418–2428, Nov. 2015. Conference Name: IEEE Journal on Selected Areas in Communications.

[18] P. Pivato, L. Palopoli, and D. Petri, "Accuracy of RSS-Based Centroid Localization Algorithms in an Indoor Environment," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, pp. 3451–3460, Oct. 2011.

[19] R. K. Yadav, B. Bhattarai, H.-S. Gang, and J.-Y. Pyun, "Trusted K Nearest Bayesian Estimation for Indoor Positioning System," *IEEE Access*, vol. 7, pp. 51484–51498, 2019. Conference Name: IEEE Access.

[20] J. Miranda, R. Abrishambaf, T. Gomes, J. Cabral, A. Tavares, and J. Monteiro, "Path Loss Exponent Analysis in Wireless Sensor Networks: Experimental Evaluation," July 2013.

[21] F. Thomas and L. Ros, "Revisiting trilateration for robot localization," *IEEE Transactions on Robotics*, vol. 21, pp. 93–101, Feb. 2005. Conference Name: IEEE Transactions on Robotics.

[22] F. Palumbo, P. Barsocchi, S. Chessa, and J. Augusto Wrede, "A stigmergic approach to indoor localization using Bluetooth Low Energy beacons," Aug. 2015.

[23] M.-Y. Jiang and Y.-D. Wang, "Localization Algorithm Research Based on the Least Square Method and Modifying the RSSI Weighted Centroid Algorithm," *Journal of Computers*, vol. 28, no. 6, pp. 269–276, 2017.

[24] J. Blumenthal, R. Grossmann, F. Golatowski, and D. Timmermann, "Weighted Centroid Localization in Zigbee-based Sensor Networks," vol. 2007, pp. 1–6, Nov. 2007.

[25] S. Subedi and J.-Y. Pyun, "Practical Fingerprinting Localization for Indoor Positioning System by Using Beacons," *Journal of Sensors*, vol. 2017, p. e9742170, Dec. 2017. Publisher: Hindawi.

[26] S. Xia, Y. Liu, G. Yuan, M. Zhu, and Z. Wang, "Indoor Fingerprint Positioning Based on Wi-Fi: An Overview," *ISPRS International Journal of Geo-Information*, vol. 6, p. 135, Apr. 2017.

[27] M. Woolley, "Bluetooth Core Specification v5.1 - Feature Overview," Dec. 2020.

[28] A. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, pp. 31–44, Mar. 1996. Conference Name: Computer.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015. Number: 7553 Publisher: Nature Publishing Group.

[30] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow - Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2 ed., 2019.

[31] I. Kouretas and V. Paliouras, "Simplified Hardware Implementation of the Softmax Activation Function," in *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1–4, May 2019.

[32] P. Warden and D. Situnayake, *TinyML - Machine Learningg with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, Inc., first edition ed., Dec. 2019.

[33] K. D. Abeyrathna, O.-C. Granmo, L. Jiao, and M. Goodwin, "The Regression Tsetlin Machine: A Tsetlin Machine for Continuous Output Problems," *arXiv:1905.04206 [cs, stat]*, June 2019. arXiv: 1905.04206.

[34] J. Lei, A. Wheeldon, R. Shafik, A. Yakovlev, and O.-C. Granmo, "From Arithmetic to Logic based AI: A Comparative Analysis of Neural Networks and Tsetlin Machine," in *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4, Nov. 2020.

[35] K. D. Abeyrathna, B. Bhattarai, M. Goodwin, S. Gorji, O.-C. Granmo, L. Jiao, R. Saha, and R. K. Yadav, "Massively Parallel and Asynchronous Tsetlin Machine Architecture Supporting Almost Constant-Time Scaling," *arXiv:2009.04861 [cs]*, Feb. 2021. arXiv: 2009.04861.

[36] A. Phoulady, O.-C. Granmo, S. R. Gorji, and H. A. Phoulady, "The Weighted Tsetlin Machine: Compressed Representations with Weighted Clauses," *arXiv:1911.12607 [cs, stat]*, Jan. 2020. arXiv: 1911.12607.

Appendix

A Hardware Components

Table A.1: Meshtech products used in our test bed.

Product	Role
MT AirMesh Mini	Gateway
MT AirMesh 1011	Beacon
MT Wristband 211	Node

UiA University of Agder
Master's thesis
Faculty of Engineering and Science
Department of Information and
Communication Technology