Delft University of Technology
Master of Science Thesis in Embedded Systems

Indoor Localization for Efficient Bike-Sharing Management

Berend Visser

TU Delft Delft University of Technology

Indoor Localization for Efficient Bike-Sharing Management

Master of Science Thesis in Embedded Systems

Networked Systems group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Berend Visser
b.visser-3@student.tudelft.nl
berend__visser@outlook.com

22th January

Author
Berend Visser (b.visser-3@student.tudelft.nl)
(berend_visser@outlook.com)
Title
Indoor Localization for Efficient Bike-Sharing Management
MSc Presentation Date
30th January 2024

Graduation Committee

dr. R. R. Venkatesha Prasad (direct supervisor)	Delft University of Technology
dr. Q. Song	Delft University of Technology
ir. G. M. Willemsen	Skopei

Abstract

In response to the growing demand for sustainable transportation solutions, bike-sharing systems have gained prominence in supporting an eco-friendly means of commuting. Within this landscape, Skopei, a forward-thinking company specializing in innovative sharing propositions, has developed a smart bike lock capable of autonomous rentals and returns.

This master's thesis delves into the application of radio frequency-based distance measurements to create an indoor positioning system for the efficient management of bike storage. Specifically, the system is designed to determine whether bikes have been correctly parked at docking locations, enabling users to conclude their rentals autonomously. The architecture of this system uses a network of anchor nodes (known-location routers) that should ascertain the positions of mobile nodes (bikes) within the bicycle storage area.

Notably, the solution developed in this thesis employs sophisticated distance measurement techniques, including frequency hopping and phase shift analysis. By finding the amplitude and phase shift over multiple frequencies, we can find the channel impulse response and estimate the distance using machine learning. We employ a novel Multi-layer Perceptron neural network regressor to improve the accuracy in the presence of complex environmental factors in bike storage environments.

In the bike storage test case, we achieved a mean absolute error in position estimation of $1.68m$ compared to $3.80m$ of a naive approach. We improved the parking state classification from 75.99% of a naive approach to 98.09% with our machine-learning-based approach.

This thesis underscores the importance of cutting-edge distance measurement methods and real-world field studies in advancing indoor positioning systems, specifically for smart bike storage management. By bridging the gap between technology and sustainable transportation, this work aims to make urban bike-sharing systems more scalable, efficient, user-friendly, and environmentally conscious.

Preface

This master's thesis concludes my years as a student. After finishing my bachelor's degree at the University of Twente,looking for a challenge, I moved to Delft for my master's degree at the TU Delft. Doing my thesis at Skopei was a very instructive experience. I could apply my academic knowledge and develop my skills in context of working with a company.

I want to deeply thank my supervisors at TU Delft, Sujay Narayana, and Dr. Rango Rao Venkatesha Prasad for their support, guidance, and advice during the project. Furthermore, I thank Niels Jacobs for enabling me to do my thesis with Skopei. I also want to thank Maurice Willemsen, my daily supervisor at Skopei, for his help during this thesis project. Finally, I want to thank all my friends and family for their support during the thesis.

Berend Visser

Delft, The Netherlands
22nd January 2024

Contents

Chapter 1

Introduction

In response to the growing demand for sustainable transportation solutions, bike-sharing systems have become an eco-friendly means of commuting. The first bike-sharing systems date back to as early as 1965, by the Witte Fietsen located in Amsterdam [13].

Recent technological developments have made smart locks possible, removing the need to exchange a lock-specific physical key and simplifying the bike rental operation. This enables autonomous operation of the bike rental locations and better throughput of bikes as rentals can be stopped and started in parallel, eliminating waiting queues for an operator.

Skopei is a company that provides sharing solutions for transportation and buildings. The product central to the thesis is a battery-powered bike lock that wirelessly connects to a local network in a designated bike storage. Registered customers can rent a bike by presenting a valid NFC tag to the lock. The lock will attach itself to the wireless network and ask a server if the user is authenticated to start a rental. The NFC tag of the user is now the key to unlock the bike. When the user returns the bike to the bike storage and locks it, the bike reconnects to the wireless network after locking. The rental is terminated automatically, and the user's key is removed from the bike lock.

The server and bike locks communicate wirelessly via Thread, an open-source IPv6-based mesh networking technology for low-power Internet of Things applications [22]. The network topology in Thread is dynamic and self-regulating. Routers are grid-powered and act as a gateway between the global internet and the local network. The routers provide connection coverage for the entire bike storage area and continuously listen for messages from the bikes. The bike locks are sleepy-end-devices, meaning they only wake up periodically; this minimizes power usage.

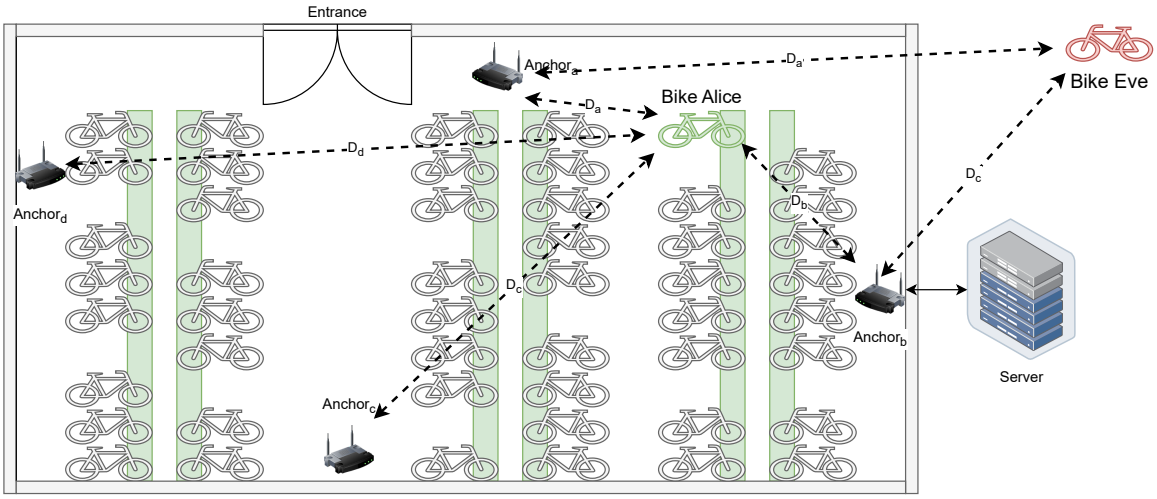


Figure 1.1: **Bike storage overview**

Consider the following scenario depicted in fig. 1.1:

Alice commutes daily to Delft Central Station by train. Using their NFC-enabled card, they start a bike rental at Delft Central Station to cover the final distance to their company's office. After a day of hard work, they return the bike (displayed in green) by parking it in a designated bike storage area in Delft Central Station. The bike connects to the wireless network and requests the server to terminate the rental.

Eve also commutes to Delft Central Station but is in a hurry. She doesn't want to miss her train and parks the bike outside (displayed in red) outside of the station. Eve is smart and knows her rental will still terminate as the range of the network also stretches outside of the station. She now doesn't have to pay extra and is almost ensured she can get the same bike again the next morning

The scenario described above clearly exemplifies the need for a localization system for the autonomous bike-sharing solution.

Integrating a localization service into such a bike-sharing system offers several benefits. It allows users and operators to easily locate the available bikes outside and in bike storages, encouraging more ridership. Bikes are an eco-friendly means of transportation. More usage reduces the need for cars, reducing traffic congestion and carbon emissions. It enhances security by tracking the bikes' location. This discourages theft, providing a more secure, lower cost, and maintainable service. Localization enables data-driven decisions, providing insights into user behavior and route popularity and optimizing station placement, pricing, and operations. Lastly, a ubiquitous localization system gives a competitive advantage in a growing market for sustainable transportation solutions.

1.1 Challenges

Implementing a localization system for the autonomous bike-sharing solution in an indoor bike storage environment described in the scenario presents several significant challenges due to the specific requirements and constraints of the system. These challenges include:

- **Signal Propagation:** Overcoming multipath fading effects due to complex signal propagation in an indoor environment
- **Localization Accuracy:** Ensuring high localization accuracy for reliable bike positioning.
- **Real-Time Localization:** Providing low-latency responses to meet user demands for real-time access to bike location information.
- **Power Efficiency:** Designing a power-efficient system to extend the battery life of bike locks and IoT devices.
- **Dynamic Environment:** Adapting to dynamic bike storage layouts and configuration changes.
- **Scalability:** Accommodating a growing number of bikes and storage locations while maintaining accurate localization.
- **Data Management:** Efficiently transmitting and processing the significant amount of location data generated by the system.
- **Competitive Market:** Meeting user expectations and contributing to a competitive advantage in the bike-sharing market.

1.2 Problem statement

Skopei does not locate the bikes when connected to the wireless network inside the bike stations. There are many ways to address the localization problem (see chapter 2). Global Navigation Satellite System (GNSS) receivers add power consumption and significant hardware cost and, due to signal attenuation and multipath effects, are not precise in indoor environments [30].

Hence, another solution is needed, which is addressed in this Master Thesis. We pose the following problem statement:

Design and implement an efficient indoor bike localization system within a bike storage environment subject to significant interference and multipath fading, utilizing Commercial Off-The-Shelf (COTS) IoT devices

1.3 Proposed solution

To tackle this problem, we propose *SkopeiLocate*. *SkopeiLocate* is specifically designed to be used with the hardware available on the bike lock; it can be integrated with the existing system through a firmware update.

SkopeiLocate is a full end-to-end solution for gathering measurement data, estimating the distance between node pairs, finding the position of the target node, and classifying if it was parked correctly.

The distance measurement reports are gathered using the Nordic Distance Measurement toolbox (DMT) [34]. The report data consists of Inphase and Quadrature measurements gathered using the active reflector multicarrier phase difference principle [38].

We implemented the system upon a Thread-based network infrastructure for scheduling distance measurements and sending the results to a base station for storage and processing.

The distance estimator is based on a multilayer perceptron(MLP) neural network trained on environment-specific measurement data. The MLP model can learn non-linear behavior, making it

suitable for estimating distances in complex multipath environments depending on many factors. MLP can be designed with an embedded friendly memory footprint while still allowing training on arbitrarily sized measurement datasets. Related works either use Weighted k-Nearest Neighbors algorithms with relatively small datasets [52] or offload model inference to a more powerful base station, which puts extra load on the Thread network [45].

We compare our estimator's performance with the $ifft_{dist}$ [1] estimator provided by the DMT. In the open-field experiment, the mean absolute error (MAE) of the raw $ifft_{dist}$ is 0.801 m, compared to 0.349 m and 0.401 m for the bias-compensated $ifft_{dist}$ and our MLP estimator respectively. In the bike storage environment the MAE's are 2.549 m, 1.805 m and 1.464 m, respectively the raw $ifft_{dist}$,the bias compensated $ifft_{dist}$ and our MLP estimator.

Every measurement gets (besides a distance estimate) a confidence estimate from $[0, 1]$ based on signal-to-interference-plus-noise ratio indicators, RSSI thresholds, and measurement Quality Indicators.

The set of *distance confidence* pairs might contain repetitions for the same node pair; the position estimator first aggregates all distance confidence pairs into a single distance confidence for every node pair. We define an error function that assigns an error to a position estimate based on the set of *distance confidence* estimations between the lock and every anchor. Then, we do a brute-force grid search, calculating an error for every position in a bounded area. Finally, we check if the position is inside the predefined allowed parking area and return the result. The position estimator has an MAE of 4.02 m using a naive approach, *SkopeiLocate* has an MAE of 1.67 m.

1.4 Contributions

We provide *SkopeiLocate* a novel machine learning-based approach to the indoor localization problem under mixed line-of-sight conditions; the key contributions are:

- **Comprehensive Indoor Localization Solution:** Developed an end-to-end system for precise indoor localization, tailored for enhancing bike-sharing management efficiency.
- **Integration with Commercial Devices:** Seamlessly integrated the system with readily available commercial off-the-shelf devices, demonstrating its practical applicability and ease of adoption.
- **Innovative Distance Estimation Approach:** Implemented a novel Multilayer Perceptron-based distance estimation method to address challenges in complex multipath fading environments.
- **Comparative Performance Evaluation:** Conducted a thorough evaluation of the proposed distance estimation technique against existing IFFT methods, showcasing its effectiveness and robustness.
- **Position Estimation Algorithm:** Created an advanced position estimation algorithm that considers environmental factors, such as walls, for enhanced accuracy in varied settings.
- **Real-World System Validation:** Validated the entire solution in a realistic environment, proving its viability and effectiveness in real-world scenarios.

[1] Based on the same raw measurement available provided to use by the Nordic Distance Toolbox, the source code for the estimation is closed source

1.5 Organisation of the thesis

The thesis is structured so that every chapter considers a subpart of the full solution. It is structured as follows:

- **Chapter 2:** Provides foundational concepts and reviews relevant literature.
- **Chapter 3:** Describes the proposed system's overall architecture and initial setup.
- **Chapter 4:** Details the measurement methodologies and data collection processes.
- **Chapter 5:** Explains the preprocessing steps for the collected measurement data.
- **Chapter 6:** Describes the design of the distance and confidence estimation.
- **Chapter 7:** Describes the algorithms and processes for estimating the position.
- **Chapter 8:** Covers the practical implementation aspects of the system.
- **Chapter 9:** Describes the evaluation of the results of the subsystems and the system as a whole.
- **Chapter 10:** Summarizes the findings and suggests future research directions.

Chapter 2

Background Theory and Related Work

This chapter provides the reader with the necessary background knowledge to understand how our proposed positioning system works. Furthermore, we provide the latest related research work.

2.1 An introduction to positioning systems

People use positioning systems in their everyday lives for a long time. When traveling, navigation on the road is possible by using road signs. Navigation systems on your phone, like Google Maps (or any similar system), usually make the job trivial. The system knows your location on the map and calculates an efficient route towards the desired destination. The phone has several methods to determine your location. Zangenehnejad et al. [63] propose that th most common way to find the location is through a Global Navigation Satellite System GNSS, the general term for systems like GPS, Galileo, GLONASS, and BeiDou. These systems have a constellation of satellites orbiting the Earth that transmit signals that our phones can pick up. The distances between the phone and the satellites can be found using these signals. Through a multilateration process, the position of the user can be found.

Figure 2.1 shows an example of trilateration. Trilateration is a case of multilateration where 3 points and 3 distances are used to find a target in a 2D space.

In this situation, the location of points A, B, and C are known. Their respective distances: d_A, d_B, and d_C. The location of the target point P is at the intersection of the three circles.

In the case of GPS, the medium used for determining the distances is a radio frequency signal of 1575 MHz. Media like visible light or acoustic waves can also determine the distance between a target and anchor points. The choice of medium depends on the use case of the positioning system. In the case of GNSS, acoustic signals are unsuitable due to the lack of an atmosphere to propagate the acoustic signal over. For underwater systems, acoustic signals propagate much farther than electromagnetic signals and, therefore, might be the better choice.

2.2 Existing bike localization

Xu Yang et al. [58] propose a system to track bikes using the assistance of the user's phone. Using the three-axis accelerator, they detect if the user has finished their bike ride. The phone's GPS location will be the bike's return location. This method could be applied to any bike, even a regular

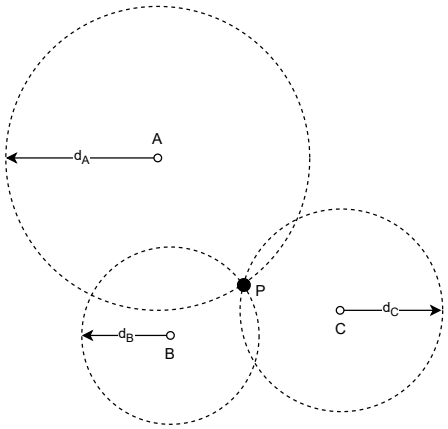


Figure 2.1: **Trilateration with anchor points (A, B, and C) and target point P**

bike. Still, it depends on the user's phone GPS, which might not work in bike storage. BikeLoc [29] uses a set of 3 antennas attached to the spinning wheel to simulate Synthetic Aperture Radar and off-the-shelf WiFi devices. This approach would require additional hardware attached to the wheel, and they use WiFi, which usually has higher power usage than low-energy protocols like Bluetooth or Thread.

2.3 Signaling technologies

According to Duran et al. [15], the starting point for any positioning system is gathering position-related signal parameters that we can use to determine the target device's location. The choice of technology to gather these parameters depends on system requirements regarding accuracy, power usage, setup time, hardware cost, etc. As for this project, the hardware is fixed, and thus, the options are limited to a subset of the solutions listed below. However, it is still important to explore these technologies as they provide valuable insight in solving the positioning problem.

2.3.1 Acoustic

Acoustic positioning systems use sound waves to determine their location. The relatively slow propagation speed of sound makes accurate systems possible.

Wang et al. [56] uses a drone that is equipped with a speaker to transmit acoustic pulses received by a microphone array on a landing platform. Due to propagation attenuation and propellor interference, the signal-to-noise ratio of the pulses received is low. Wang et al. use a Matched Filter Tree to convert the pulse detection to a tree search problem. This allows them to overcome the non-linear Doppler distortion. Chen et al. [10] presents an underwater positioning system. The pairwise distances are found between divers to overcome the limited propagation range in an aquatic environment. The pairwise distances are used to determine the relative positions of the divers. Chen et al. also present methods to overcome the rotational and flipping ambiguities using a binary classifier.

Other works combine acoustic signals with other signal sources. Fischer et al. [16] use Bluetooth's Received signal strength indicators (RSSI) to estimate and calculate course grain distance. The accuracy of this method is very limited due to the high variability in RSSI. This is overcome by extending the system with Ultrasonic Time Difference Of Arrival measurements.

Luo et al. [28] use the speaker and microphone of the same device to achieve echolocation and combine it with the inertial measurement unit(IMU) to do Simultaneous Localization and Mapping (SLAM). First, a map of the environment is built by collecting measurements of a user traversing the environment. The acoustic echo data is used to find loop closures and overcome the drift problem of dead-reckoning from the IMU.

2.3.2 Magnetic

The earth's magnetic field can also obtain positional information without deploying additional hardware in the area. Anomalies in the magnetic field induced by walls and floors make direction-finding in dead reckoning systems challenging. Magicol [48] uses the anomalies to build a magnetic field map with a regular radio map to achieve improved accuracy over just using a radio map. They also show that, albeit at the cost of lower accuracy, just using the magnetic field map has a 9 times lower power usage compared to WiFi-based tracking. Wang et al. [55] use magnetic field maps to locate cars in tunnels where GNSS signals are unavailable. The system was tested for 36 months in 56 tunnels and showed accurate results using a heterogeneous set of off-the-self smartphones, which makes for an affordable solution. Kok et al. [24] also use the spatially varying magnetic field and give a framework for incorporating it with other signals from the IMU for SLAM.

2.3.3 Radio frequeny

Radio Frequency (RF) indoor localization is a technology used to determine positions inside buildings where GPS is ineffective. It utilizes the radio waves from devices like smartphones or RFID tags. The system measures these waves' signal strength or travel time to triangulate the device's location. This technology is valuable for navigation, warehouse asset tracking, and enhancing retail and security systems in complex indoor spaces like malls or airports. RF indoor localization is a key aspect of indoor positioning systems, vital for improving efficiency and user experience in indoor environments.

Ultra-Wide-Band

According to Gezici et al. [17], Ultra-wide-band (UWB) signals have a high time resolution and can provide centimeter-accurate position estimation. Zand et al. [62] use UWB with time-difference of arrival (TDOA) and provide a solution that overcomes the strict clock synchronization required in similar methods. The precision of the UWB system is highly affected when there are Nonline-Of-Sight(NLOS) measurements. [8] solves this by applying machine-learning techniques like feature-based Gaussian distribution to detect these NLOS samples.

2.3.4 WiFi

In urban environments, the advertisement messages of (sometimes many) Wifi access points (AP) can be detected and used for localization. Youssef et al. [59] use the RSSI of existing Wifi communication infrastructure to create a map of fingerprints (radio map) for all locations within a certain area in the offline phase. In the online phase, the user's location is determined by matching the RSSI of access points to the radio map. Generating the radio map can be a time-consuming problem. Rai et al. [39] try to automate this process by letting users scan for wifi signals and track their movement with IMU data. A map of the walls and pathways in the environment is the only input required.

Rizk et al. [41] train a deep-learning-based localization model by fusing RSSI and Round-trip-time (RTT) measurements to extract high-level features using deep canonical correlation analysis. The limitations in the fingerprinting technique are overcome by combining it with time-based techniques like RTT, but many measurements are required to train the algorithm.

2.3.5 RFID

Many stores employ anti-theft systems using low-cost radio-frequency identification (RFID)tags attached to valuable items. This positioning system checks device presence in specific points of interest, like a store exit. Saab et al. [42] track the position of a device containing an RFID-tag reader by scanning for low-cost passive RFID tags placed in the environment. ReLoc [27] uses the RSSI and phase of the RFID tags to find the relative position using a scanner that traverses a warehouse.

2.3.6 Bluetooth and 802.15.4

Like WiFi, which falls into the category of Wireless LAN, Wireless personal area networks can be employed to gather positional signals. The hardware available (BMD-340 [53]) contains a Bluetooth and 802.15.4-enabled chip. Proprietary protocols are also available, which are expanded on later. Puspitaningayu et al. [37] propose an 802.15.4-based radio map method using parameter optimization to assign multiple fingerprints. Bencak et al. [4] propose an Indoor Positioning System (IPS) based on Bluetooth Low Energy (BLE). Beacons are deployed in a test area, and the receiver carries a BLE receiver to detect passing-by.

2.4 Positioning-related parameters

Different parameters can be obtained depending on the signal technologies to gain information about the position.

2.4.1 RSSI

Information about the received signal strength is commonly available in wireless technologies like Bluetooth and WiFi. The RSSI is usually expressed in decibel milliwatt dBM. It comprises the average power of a message received at the antenna. It can be used to estimate the distance between two nodes, as signal power depends on the distance. Kumar et al. [26] propose to use the Friiss Free Space Path model to estimate the distance:

$$RSSI = RSSI_{d0} - 10n \log(\frac{d}{d_0}) \tag{2.1}$$

Here, $RSSI_{d0}$ is the RSSI value at some reference distance, N is the environment-specific path-loss exponent, and d is the current distance. This formula can be inverted to find a distance estimate for a specific RSSI measurement. This approach is simple and low-cost, but it suffers from poor accuracy due to high variability in RSSI values due to multipath and NLOS [47],[19].

2.4.2 Angle of arrival

When sound arrives at our ears, depending on the direction, there will be a slight difference in arrival time at each ear. This effect enables us to detect from which direction the sound is coming. Chen et al. [9] use this effect for positioning by measuring this phenomenon electronically. On the left in figure 2.2, we see that node P sends an RF signal to node A with 2 antennas. The distance the signal travels depends on the direction of P; thus, the signal experiences a different phase shift for each antenna that can be measured. The phase between the receive the antennas, ϕ_{a1} and ϕ_{a2} can be represented by:

$$\phi_{a1} - \phi_{a2} = \frac{2\pi d \sin\theta}{\lambda} \tag{2.2}$$

In 2.2 d is the distance between the antennas, θ the angle of incidence, and λ the wavelength. Thus, the angle of incidence can be found by knowing the phase difference between the antenna. The clock rate and phase must be synchronized between the antennas. This can be achieved using the same oscillator source and multiplex between antennas.

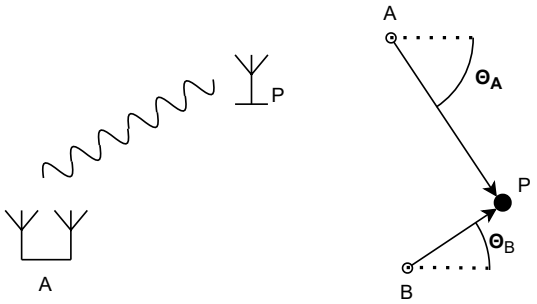


Figure 2.2: **Angle of arrival measurement (left) and triangulation (right)**

The triangulation process is depicted on the right in figure 2.2. The positions of node A and B are known, and target P lies at the intersection of the two lines. SpotFi [25] determines the AoA with already deployed WiFi Access points with only 3 antennas. It is tested in a multipath-rich environment and assigns likeliness values to each found path.

2.4.3 Time of arrival

The distance between two nodes is calculated using a signal's propagation delay. Bensky [5], the distance can be found by:

$$d = (t_2 - t_1)c \tag{2.3}$$

In equation 2.3, d is the distance, t_2 is the time of arrival at the receiving node, t_1 is the time of transmission at node 1, and c is the propagation speed. The propagation speed of light is $3 \cdot 10^8[\frac{m}{s}]$ so it travels 30 cm every nanosecond, which means that very precise time measurements are required to get an accurate estimation of the distance. When sound is used as a signal medium, the travel speed is much lower, and more precise measurements are easier to achieve. The clocks between the sender and receiver need to be precisely synchronized. Alternatively, the round-trip time can be calculated. Node two responds to a message initiated by node one, removing the need for node synchronization. When node 2 receives the initial message, there is some processing delay before it can send a response. We need to remove this processing delay to find the total propagation delay of both messages. Abedi et al. [1] track users in a building by doing round-trip time measurements. It exploits the WiFi protocol by sending connected nodes a fake message, which they respond to with an ACK. They use the ESP32 microprocessor with a clock frequency of 240 MHz, which enables timestamping within a few nanoseconds. They achieve accuracy by averaging many measurements.

2.4.4 Phase of arrival

Similar to the time of arrival, the phase of arrival can also be used to determine the distance between two nodes. The phase of a received signal depends on the distance the signal has to travel via the following relation [5]:

$$d = \frac{c}{f}(\frac{\theta}{2\pi} + n) \tag{2.4}$$

Here d is the distance between the nodes, c is the propagation speed through the medium, f is the frequency, θ is the phase, and n is an integer number of wavelengths. The combination of n and θ is the total number of wavelengths that fit between the nodes. The corresponding distance to a phase measurement is ambiguous as the same phase will be measured by adding N times one wavelength. The ambiguity can be removed by measuring the phase for a different frequency:

$$d = \frac{c}{2\pi} \cdot \frac{\theta_2 - \theta_1}{f_2 - f_1} \tag{2.5}$$

In 2.5, we get the phase slope by taking the difference between θ_2 and θ_1 and dividing by the difference in frequency between f_2 and f_1. This is under the assumption that the distance is less than $\frac{c}{\Delta f}$, which is 300 m for a $\Delta f = 1[MHz]$.

As with AoA, the oscillators must synchronize between receiving and transmitting devices to measure a phase offset relative phase offset. Alternatively, the receiver and transmitter can switch roles and measure the phase shift with an unsynchronized local oscillator on both ends [36], [62]. The phase offset can then be calculated by:

$$\theta_{offset} = (\theta_{lo1} - \theta_{lo2} + \theta_{prop}) + (\theta_{lo2} - \theta_{lo1} + \theta_{prop}) = 2 \cdot \theta_{prop} \tag{2.6}$$

In the unsynchronized case, a device measures the phase offset induced by signal propagation θ_{prop} plus the difference between its local oscillator and the remote node, θ_{lo1} and θ_{lo2}. Adding the two measurements cancels the offset of the local oscillators, and we are left twice the propagation offset.

2.5 Machine learning

A machine learning model can also estimate the distance between two nodes based on a set of position related parameters.

In statistical learning, the work "Introduction to Statistical Learning" of James et al.[20] provides a foundational understanding of the concepts and methodologies.

Complementing this, the study by Zhang et al. [64] focuses on enhancing accuracy in a specific application area. They show improved ranging accuracy by employing machine learning. They combine AoA with RSSI and train their Artificial neural network with simulated channel data and actual measurements.

Nessa et al. [31] provides an overview of machine learning for indoor positioning. They state that machine learning for indoor positioning is still in its infancy, and standardization for training data is required.

2.6 Positioning algorithms

Positioning systems can be distributed or centralized. For a centralized system, all measurement data is sent to a central server. This increases the overhead in data exchange between servers and end devices. Still, the upside is that all data goes through a central point, allowing for more control and processing power, compared to the distributed case where end devices do all processing.

We can also divide between fingerprinting ([11], [65], [32]) , range-free ([3]), and ranged systems ([45], [1], [51]). No absolute distance between anchors is used in range-free systems, but the connections between nodes or the amount of hops between nodes. This usually relies on high node density.

For fingerprinting methods, measurements are compared to a prerecorded database containing records for every possible location, requiring a long initialization time. The last category is the ranged systems, where distances or angles between nodes are estimated to find the position.

Systems can be active, where measurements are explicitly initiated between nodes, or passive, where a node measures background signals that are available independently from the positioning system. In fig. 2.3, a taxonomy overview is given for indoor positioning systems. We have three main

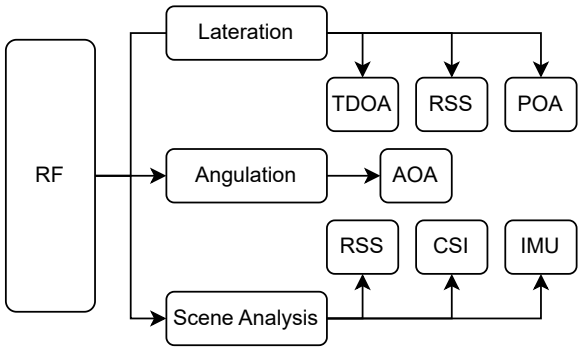


Figure 2.3: **Taxonomy of radio frequency based indoor position systems**

methods to find the position: *lateration*, *angulation*, and *scene analysis.* In lateration, we estimate the distance based on the positional-related parameters using one of the aforementioned techniques. Using a set of distances, we estimate the position.

In angulation, we try to estimate the angle between two nodes, and using multiple angles, the position can be estimated to be at the intersection.

In scene analysis, we gather fingerprints for locations of interest, and when we estimate the position, we look up which point fits best.

Chapter 3

System Overview and Initialization

In this chapter, we describe the localization solution *SkopeiLocate*. Figure 3.1 gives a top view of a bike storage. The goal is to provide localization of bikes for fleet management for a bike storage facility.

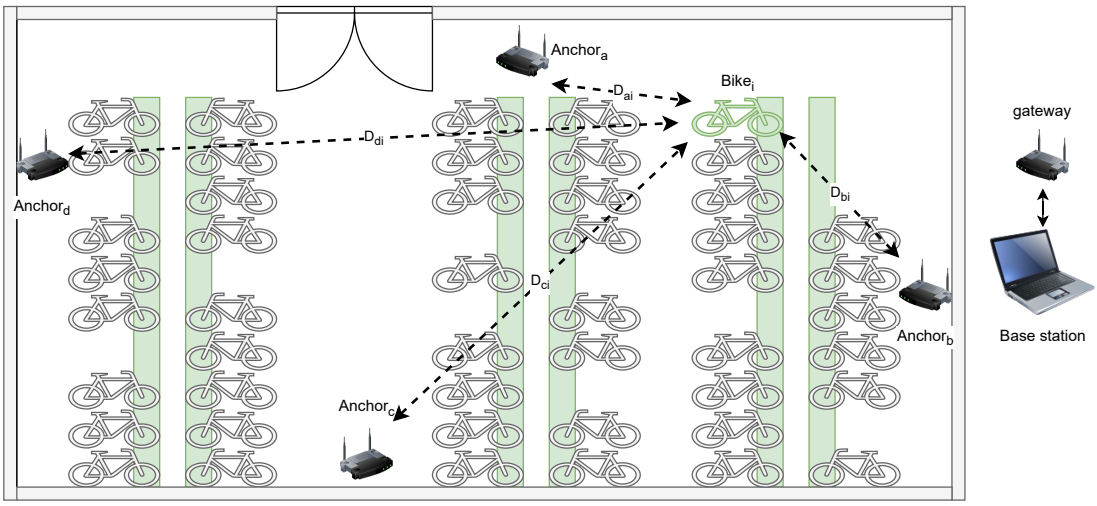


Figure 3.1: **Top view of a bike storage, with Anchor Nodes a, b, c and d and Bike i and distances D_{ai},D_{bi},D_{ci} and D_{di} and a gateway connected to a base station**

The anchors provide a wireless network covering the bike storage displayed in fig. 3.1. The green bike has a smart bike lock that can connect to the anchor. After connecting, it will measure the distance between itself and all the anchor nodes. The base station is also connected to the network and will receive the measurements' results.

Figure 3.2 shows a functional overview of *SkopeiLocate*. The first step of the algorithm is to initialize parameters specific to the bike storage, like walls and anchor node positions. After initialization, the system is ready to receive requests to localize bikes. The localization requests are initiated by some event, for example, after a user has parked their bike and tries to terminate their bike rental. For evaluation purposes, we trigger these localization requests manually. When a request is made to localize the bike, the first step is to take range measurements. In this stage, we measure range-

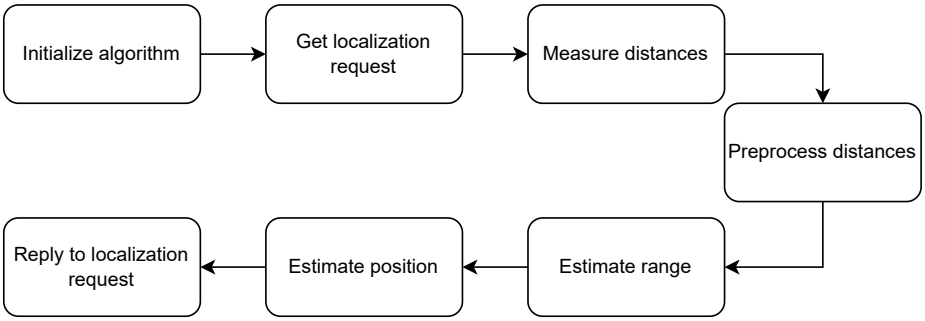


Figure 3.2: ***SkopeiLocate* system overview**

related parameters, like signal strength between a pair of devices. We use a centralized localization system, meaning all results are processed on a central server. This approach has the advantage of having the central server, or base station (BS), fully control the positioning process. This reduces bikes and anchor nodes to only being a measurement device while moving all complex logic, like signal processing and localization, to a more computationally powerful device.

Next, the measurement data goes through a preprocessing stage. In the processing stage, the raw data from the measurements is converted to data that the range estimator can directly use.

The range estimator converts the pre-processed measurements into a distance and confidence estimation. The pre-processed data consists of many features, which all can have an impact on what the distance is. The goal is to estimate a distance based on these features and output a distance closest to the actual distance. Not all measurements are equal, and we want the subsequent positional algorithm to account for this. Hence, based on the feature set, we also estimate how confident we are that this distance is what we predicted.

The range estimator takes the measurement set and treats every measurement as a separate independent entity. During a measurement, the bike is assumed to be locked, so the position can also be constant. The position estimator attempts to find a position that best fits the measurement set. After a position has been found, we classify if this position is within the area the fleet operator has categorized as a valid parking space. Finally, we reply to the localization request by sending our results consisting of a position and correctly parked boolean.

In the following chapters, we will explain every subsystem of the localization solution in detail.

3.1 System initialization

We see an overview of the initialization steps in fig. 3.3. We will go through each of these steps in the following sections.

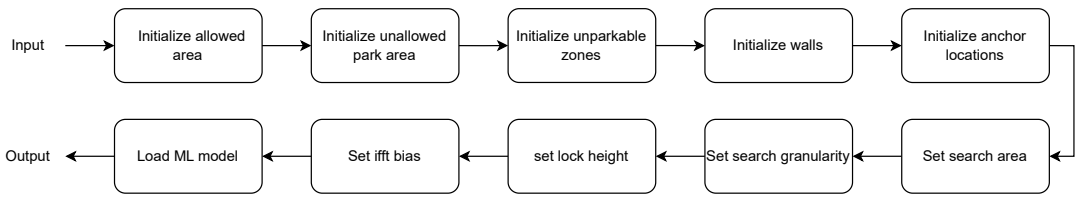


Figure 3.3: **Initialization procedure**

Every location where the *SkopeiLocate* operates is different, and we need to provide specific parameters for every location to *SkopeiLocate* to operate optimally. We can gather many location-specific parameters, but measuring can be time-consuming; therefore, we need to make a compromise in selected parameters. The goal is to find a set of parameters that is easy to measure and has maximum benefit for the positioning algorithm. Our set of parameters consists of the following items:

- Position and orientation of the Anchor node
- Position and orientation of the Lock node (only for reference)
- Attached height and orientation of the Lock
- Position and material of the walls
- Search area
- Unparkable area
- Allowed parking area
- Bias factor
- ML model

The position of the lock and anchor nodes is a fundamental feature, and we use it to calculate the reference distance between node $i : (x_i, y_i, z_i)$ and node $j : (x_j, y_j, z_j)$ by:

$$D_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} \tag{3.1}$$

The radiation pattern of the wireless devices is not isotropic but is directional. Combining the position and orientation allows us to find the angle between a pair of nodes. In the evaluation chapter 9, we will show the effect of lock angle distance estimation. If the radiation pattern and signal path are known, we can compensate for this non-ideal behavior in signal gain. The orientation and position of the anchor nodes are assumed to be static and thus only have to be measured once.

The bike's z component or height is assumed to be fixed. If the bike is parked civilly, it stands up with an unknown bearing (north, south, etc.), but the height of the lock is fixed. We note down the orientation of the lock, given it is standing up. The orientation of the lock can also be measured using, for example, an inertial measurement unit, but as this is a dynamic variable, it is not used in initialization.

When electromagnetic signals propagate in the real world, they are bound to meet obstructions like walls, ceilings, furniture, etc. When a signal transitions from one material to another, its party reflects and partly propagates depending on the material shape and characteristics. In wireless communication, this effect of signal echoing due to the environment is known as multipath fading. While multipath fading is complex, large static objects like concrete walls can greatly affect signal propagation. By recording the position and material of the walls, we can compensate for this effect.

The positioning algorithm takes a certain space in which it will search for the position of the lock. Usually, bike storage facilities are in urban environments directly next to, for example, office spaces. Since it is unlikely that the bike was parked in the middle of an office, we can remove these areas from our prediction set. Also, we denote where the bike is allowed to park so we can classify if it was parked correctly or not.

Finally, we provide a bias factor and a machine learning model, we will expand on these parameters in chapter 6.

Chapter 4

Measurement and Data Collection

The measurement stage is the first step after a localization request. A series distance estimation between the node pairs is required to determine the position of the target node, we explain how the position is estimated based on the measurements in section 7. In figure 3.1, the top view is given of a bike storage containing a set of anchor nodes $A = \{a, b, c, d\}$ and bike nodes $B = \{i\}$. We measure distance with n repetitions for every bike anchor pair. The measurements use the active reflector method [44]. This method has two distinct roles: one node will be the initiator, and the other will be the reflector, so either the bike or the anchor node can be the initiator, and the other is the reflector.

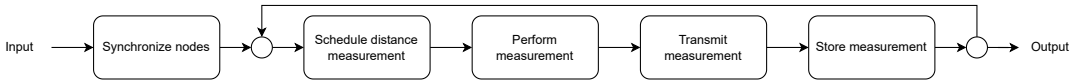


Figure 4.1: **Measurement process**

We have a pair of nodes where a is the anchor, and i is the bike. We define a matrix $\mathbf{M}$ for measurements, and M consists of two rows. In the top row, a is the initiator, and in the bottom row, b is the initiator. The columns represent repetitions of measurements. Matrix $\mathbf{M}$ then becomes:

$$M_{ai} = \begin{bmatrix} \text{Measurement}_{ai1} & \text{Measurement}_{ai2} & \dots & \text{Measurement}_{ain} \\ \text{Measurement}_{ia1} & \text{Measurement}_{ia2} & \dots & \text{Measurement}_{ian} \end{bmatrix}$$

This matrix represents the distance measurements for a single anchor bike pair, but we have a set of these pairs. We can create a matrix D, where element D_{ij} represents the matrix measurement M between anchor node i and bike node j. Let $A = \{a, b, c, d\}$ represent the set of anchor nodes, and $B = \{i\}$ represent the set of bike nodes. Matrix D then becomes:

$$\mathbf{D} = \begin{bmatrix} M_{ai} \\ M_{bi} \\ M_{ci} \\ M_{di} \end{bmatrix} \tag{4.1}$$

At the top level, we define the measurement in the following three states: **Succesfull**, **Failed** and **Not executed**. The need for the first two states is obvious; when a measurement is started, something in the process might fail, like a bitflip during transmission or connection loss, which causes the measurement to either fail or succeed. The **Not executed** state is needed to allow

measurements to be skipped, so there is no attempt to do that measurement. Since measurement takes time, some measurements might not be beneficial for the positioning stage, and hence, the measurement time can be better invested in a different measurement.

4.1 Measurement scheduling

Due to the nature of the active reflector principle, a single node can only measure with one other node at a time. This means measurement jobs between the nodes need to be scheduled, and every node can be viewed as a processor that can only process one job at a time. When we want to localize a bike, every measurement involves the target bike node, so measurements can not be performed in parallel. Measurements between exclusive node pairs can be performed in parallel, this falls outside the scope of this work. Instead, we focus on the case with a single bike node. A list of measurement jobs is created when a localization request is made. We choose a static scheduling approach where all nodes are treated equally, meaning a bike node attempts to measure with all anchor nodes for a fixed number of repetitions. This approach was chosen for simplicity.

For example, let us assume a set of anchor nodes $A = \{a, b, c, d\}$ and bike nodes $B = \{i\}$ from figure 3.1, where we take 6 repetitions for every measurement. Let M_{all} be a set of all measurements. Each measurement is indexed by the anchor node i, the bike node j, and the repetition k. Note that the order of i and j depends on which node acts as the initiator. The total number of measurement in M_{all} is:

$$M_{all} = \sum_{i \in A} \sum_{j \in B} \sum_{k=1}^{6} Measurement_{ijn} + \sum_{i \in A} \sum_{j \in B} \sum_{k=1}^{6} Measurement_{jin} = 48$$

Here's what each part of the summation represents:

- The outer summation ($\sum_{i=1}^{4}$) iterates over the anchor nodes, from 1 to "i."
- The middle summation ($\sum_{j=1}^{1}$) iterates over the bike nodes, from 1 to "j."
- The inner summation ($\sum_{k=1}^{6}$) iterates over the repetitions, from 1 to "n."
- $Mabn$ represents the measurement taken when initiator node a interacts with reflector node b during repetition k.

A round-robin scheduling scheme dispatches this set of 48 measurements. Figure 4.2 shows how this scheduling works in practice. On the x-axis, we have time, and on the y-axis, we have the different anchor nodes. Since all jobs depicted involve the bike nodes, we have to schedule them one after the other. In the round-robin schedule, every task is scheduled in a circular order. The advantage of this approach is that it is simple to implement and doesn't starve specific nodes from getting any measurements.

The scheduler is running on the base station,

4.2 Measurement data

We have defined that measurement matrix $\mathbf{M}$ consists of a set of elements $Measurement_{ain}$. Here, we define what the measurement itself consists of exactly. First, the goal of each measurement is

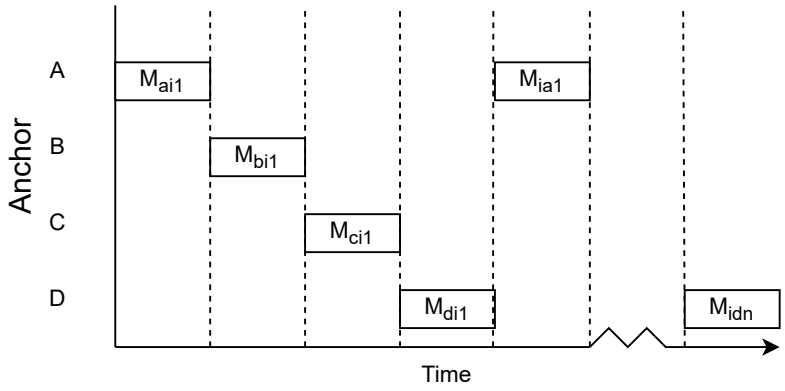


Figure 4.2: **Measurement schedule**

to find the distance between the anchor node a and the lock node i. This is achieved by measuring position-related parameters x_1 to x_n. This gives the following vector:

$$\vec{Measurement}_{ain} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{4.2}$$

Different parameters can be gathered depending on the signal medium, as explained in chapter 2. Due to the requirement constraints, we use commercial off-the-shelf hardware with a 2.4 GHz ISM-band radio. This hardware has a single antenna on both the anchor and bike nodes, making angle of arrival not an option. Also, due to the low clock frequency of 64[MHz], fine-grained time of flight measurements are not precise. The speed of light is $\approx 0.3[\frac{m}{ns}]$ and with a clock period of $\frac{1}{0.0.64} = 15.625[ns]$, we can only measure with a granularity of $15.625 \cdot 0.3 \approx= 4.68[m]$, which is under our precision requirement. Instead, we utilize the phase of arrival distance measurements. When a signal is transmitted, it experiences a phase shift depending on the distance it travels:

$$\phi = 2\pi \frac{d}{\lambda} \pmod{2\pi}$$

Our hardware can capture this phase shift between two devices over a range of frequencies spanning from 2.400 GHz to 2.480 GHz with steps of 1 MHz. This principle is referred to as multiple carrier phase difference ranging. However, to measure the phase offset between two devices directly, both devices need to be synchronized precisely. This is hard to achieve due to small frequency offsets between the oscillation crystals in both devices. This can be overcome using the active reflector principle shown in figure 4.3.

The initiator emits a constant carrier wave received at the reflector. The reflector measures the phase offset with respect to its local oscillator using an IQ measurement. The magnitude and phase of a received signal can be determined using the IQ measurements: Inphase and Quadrature. The reflector then transmits a carrier wave back to the initiator at the same frequency, and the initiator measures that signal compared to its local oscillator. In chapter 5, we detail how to extract the phase offset from the raw IQ measurements.

For every IQ measurement, we also measure the signal-to-interference plus noise ratio. This is defined in three steps: **Good**, **Medium**, and **Bad**. This metric can be used to determine whether the IQ measurement is reliable.

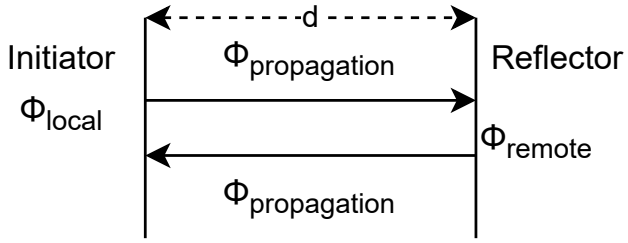


Figure 4.3: **Active reflector**

Furthermore, we obtain the Received Signal Strength Indicator (RSSI) from the initiator and reflector. The received signal strength indicator indicates the energy level of the signal received at the antenna. When a signal propagates, it is attenuated due to several factors. According to the inverse square law, the farther a signal travels in free space, the more it attenuates. Also, when a signal has to travel through materials, like a wall, it is attenuated. Finally, due to reflections on walls, the same signal might interfere with itself, affecting the energy level.

The measurements are performed at a specific point in time **t**. This parameter is used to find temporal relations in the distance measurement. We know that bike storage might be more crowded at specific times of the day, causing signals to undergo different multipath fading effects. By recording the time of the measurement, we can compensate for this.

4.2.1 Toolbox specific data

Besides the raw data described in subsection 4.2, Nordic provides an undisclosed algorithm that estimates the distance between a pair of devices based on the abovementioned raw measurement data [34]. This data consists of the following distance estimation methods:

- ifft
- phase slope
- path loss
- high precision

The *ifft* method finds the distance based on the IFFT spectrum of the signal, the phase slope method finds the distance based on the average phase slope, the path loss method uses the friis path loss, and the high precision method uses "advanced spectral analysis. In chapter 9, we will discuss the performance of these algorithms and show how we improved them.

We are also provided a quality indicator of the measurements that can take the following values: **OK**, **Poor**, **Do Not Use** and **CRC Fail**. The **CRC fail** indicates some problem with the data transmission.

4.3 Synchronization

When a measurement job is scheduled, the cooperation of both devices is required. Hence, both devices must be informed that they should start the job. Informing both nodes that they should start a measurement can be done by sending a starting signal at the right instant when the job needs to

be started. The job scheduler runs on the base station, as explained in section 4.1. Therefore, when a job is dispatched from the base station to the node, it must first traverse the wireless network.

In chapter 9, we will show that measurements take milliseconds, and the latency of transmitting messages is in the same order of time. Hence, we need to schedule jobs ahead of time to overcome this inefficiency. Besides this, when the measurement is initiated using the Nordic distance toolbox, the reflector waits actively for a message from the initiator, blocking other radio traffic of the device. If the initiator were to start 100 ms later due to some network congestion on the job start message, the reflector would wait all this time for no reason while blocking other network traffic.

This limitation is overcome by synchronizing the time on both devices with a naive round-trip-time approach; this approach is shown in figure 4.4.

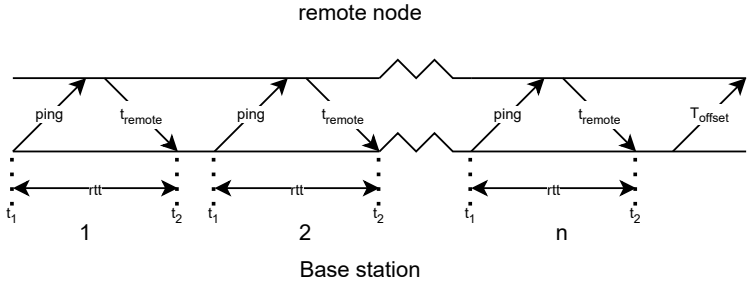


Figure 4.4: **Round-trip time**

The base station sends a ping to the remote node it wants to synchronize with and records the transmission time t_1. After some transmission delay, t_{trans1} remote node receives this message and replies to the base station with a message containing its local time. After some delay t_{trans2}, the base station receives the message from the remote node and records the time of arrival of the message t_2. We then find the offset between the devices as follows:

$$offset = \frac{(t_1 + t_2 - 2 \cdot t_{remote})}{2}$$

The round-trip-time is found as follows:

$$rtt = t_2 - t_1$$

This approach assumes that the transmission time from base station to node and vice versa is the same. This measurement is performed multiple times before setting the offset of the remote node. The advantage of multiple measurements is that the average offset over a series of measurements can be taken after filtering the outliers, leading to more precise synchronization. The outliers are filtered as follows:

$$\text{Filtered RTT} = \{x \in \text{RTT values} \mid x \leq 1.3 \times \min(\text{RTT values})\}$$

When the RTT of a measurement is higher than 1.3, the minimum value, we consider the measurement an outlier. The value of 1.3 was experimentally found to work in the test setups we tried. Since the network's communication is via a Thread network, which is a mesh network, we can have multiple hops between the base station and the end node. Every hop adds variability in the delay; a value of 1.3 is a good compromise between not filtering too many values and keeping large outliers out.

The set of filtered RTT's has a corresponding set of offsets; we find the final offset by calculating the mean of this set:

$$\text{Final Offset} = \frac{1}{N} \sum_{i=1}^{N} \text{Filtered offset}_i$$

4.4 Transmission and storage

After the measurement, the gathered data is transmitted from the measurement node to the base station. Both nodes have the same data on their local device when a measurement succeeds. However, this is not always the case, meaning the data was corrupted due to some measurement problem. Both devices in a measurement transmit their respective measurement data to the base station, allowing the base station to detect discrepancies between the data.

The OpenThread network is employed to facilitate the communication between devices. It is a low-power network, so it is ideal for battery-powered bike locks, but this comes at the cost of bandwidth. We transmit all raw measurement data unprocessed to the base station for data analysis purposes, making a single measurement just fit inside the maximum transferable unit of an Openthread network.

All measurements are stored in a database on the base station, allowing us to evaluate different algorithms on the same data without constantly waiting for new measurements, speeding up development time. When a measurement is scheduled, an entry is created in the database with a specific measurement ID. When a measurement is scheduled, the device is aware of this ID and adds it when it transmits the measurement result to the base station so the database knows where to store the result.

Chapter 5

Measurement Data Preprocessing

In the preprocessing subsystem, we take the raw measurement reports stored in the database described in subsection 4.4 and apply a series of preprocessing steps to prepare the matrix D from 4.1 for the range estimator. Figure 5.1 overviews the preprocessing steps taken. In the following sections, we explain the steps.

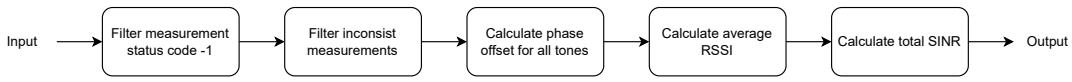


Figure 5.1: **Preprocessing steps**

5.1 Faulty measurement filtering

When a measurement is taken, it can fail due to several causes. For example, the Initiator and Reflector could not synchronize to perform a measurement, and the measurement, therefore, times out. When this happens, the anchor and bike nodes will transmit a measurement report indicating a status code of -1. We can have a case where both the initiator and reflect are successfully measured, but the data they return is incomplete. A combination of the cases can also happen, and these measurement reports are removed from the report set.

5.2 Preprocess IQ measurements

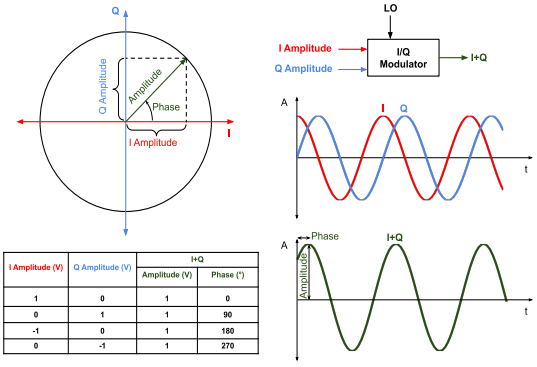
From a valid measurement report (see 4.2, we can extract the following vectors for the IQ measurements:

$$\begin{aligned}
\mathbf{Local_I} &= [I_1, I_2, \ldots, I_{80}] \\
\mathbf{Local_Q} &= [Q_1, Q_2, \ldots, Q_{80}] \\
\mathbf{Remote_I} &= [I'_1, I'_2, \ldots, I'_{80}] \\
\mathbf{Remote_Q} &= [Q'_1, Q'_2, \ldots, Q'_{80}] \\
\mathbf{Frequency_i} &= 2.4\,GHz + i, MHz
\end{aligned}$$

The signal used for finding the phase offset between the initiator and reflector can be described by the narrowband signal model:

$$x(t) = A\cos(2\pi f_c t + \theta) \cdot s(t) \tag{5.1}$$

In equation 5.1, A is the amplitude, f_c is the carrier frequency, θ is the phase offset, and $s(t)$ is the envelope function. The envelope function can be used to encode data, but we use a constant carrier for the phase measurements, so the function is 1. The goal of our measurements is to extract the phase In figure 5.2 [1], a visual representation of IQ components is given. When a carrier 2.4 GHz carrier signal is received at the antenna, it is downconverted to a lower frequency by mixing (multiplying) it with a local oscillator (LO) at the same carrier frequency. The local oscillator mixes the signal twice at a 90 deg offset. This approach allows the receiver to find the Inphase and Quadrature component of the signal. Using the IQ components, we can find A and ϕ from 5.1.



I Amplitude (V)	Q Amplitude (V)	I+Q	
		Amplitude (V)	Phase (°)
1	0	1	0
0	1	1	90
-1	0	1	180
0	-1	1	270

Figure 5.2: **IQ phasor diagram**

$$A = \sqrt{I^2 + Q^2} \tag{5.2}$$

$$\theta = \arg\left(\frac{Q}{I}\right) \tag{5.3}$$

The A and θ components depend on the path the signal travels, we will explain this in detail in section 6. Using the IQ measurements at the initiator and reflector, we can θ using equation 5.3 and amplitude A using equation 5.2. We can then find θ_{offset} by:

$$\begin{aligned} \theta_{offset} &= \theta_{init} + \theta_{refl} \\ &= (\theta_{lo_{init}} - \theta_{lo_{refl}} + \theta_{prop}) + (\theta_{lo_{refl}} - \theta_{lo_{init}} + \theta_{prop}) \\ &= 2 \cdot \theta_{prop} \end{aligned} \tag{5.4}$$

The phase offset due to signal propagation can then be extracted using equation 5.4. Note that θ_{init} consists out 2 components essentially:

- The phase offset between the LO at the initiator and the LO at the reflector
- The phase offset due to propagation of the signal

[1] IQ phasor diagram by Vigneshdm1990 is licensed under CC BY-SA 4.0

By adding the phase measurement of the reflector and initiator, the offset due to the local oscillator cancels out, and the twice propagation phase offset remains.

The vector **Phase_offsets** with 80 elements can be represented as:

$$\textbf{Phase\_offsets} = [\phi_1, \phi_2, \ldots, \phi_{80}]$$

The vector **Amplitude_initiator** with 80 elements of amplitude A, indexed as A_1 to A_{80}:

$$\textbf{Amplitude\_initiator} = [A_1, A_2, \ldots, A_{80}]$$

The vector **Amplitude_reflector** with 80 elements of amplitude A, indexed as A_1 to A_{80}:

$$\textbf{Amplitude\_reflector} = [A_1, A_2, \ldots, A_{80}]$$

5.3 Phase unroll

Sinusoidal signals have a phase shift between 0 and 2π, this is also the case for our vector **Phase_offsets**. In figure 5.3, an example is **Phase_offsets** for a short distance in *green* and a long distance in *red*. The distance of the red line is twice as far as the green line; hence, the phase slope will be twice as steep. Note that the phase of the green line starts at 1.5π. While the start of the phase depends on the distance, if the distance changes by a full wavelength, only 12.5 cm for 2.4 GHz signals, all phases will have shifted by a 2π. The initial phase also depends on the phase offset between the local oscillator of the nodes, which might shift significantly if measurements are repeated due to oscillator deviations.

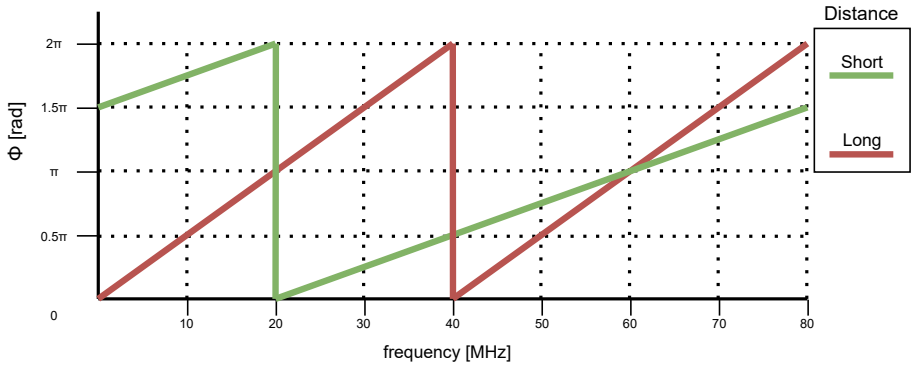


Figure 5.3: **Phase response**

Since the signal is periodic by two pi, we can add an integer multiple of 2π, without changing the signal. We see a 2π jump at 20 MHz for the green line and 40 MHz for the red line. By removing phase discontinuities and preserving the true phase progression, phase unwrapping enables precise analysis and interpretation of periodic signals, ensuring that phase-related measurements and calculations are accurate and meaningful. It involves detecting and correcting phase jumps or discontinuities, thereby revealing the underlying continuous phase evolution of the signal.

5.4 Calculate summed SINR

The measurement reports also provide a signal-to-interference-plus-noise ratio indicator for every tone for the initiator and reflector.

The vector **SINR_initiator** with 80 elements, indexed as $SINR_1$ to $SINR_{80}$:

$$\mathbf{SINR_{initiator}} = [SINR_1, SINR_2, \ldots, SINR_{80}]$$

The vector $\mathbf{SINR_{reflector}}$ with 80 elements, indexed as $SINR_1$ to $SINR_{80}$:

$$\mathbf{SINR\_reflector} = [SINR_1, SINR_2, \ldots, SINR_{80}]$$

While SINR values may vary in discrete integer steps from 0 to 2, the cumulative SINR is still a valuable metric for confidence estimation, providing insights into the overall performance and quality of the measurement.

5.5 Average RSSI

The RSSI of the initiator and reflector is given in the report. Averaging RSSI values from two devices enhances distance estimation accuracy by mitigating signal fluctuations, reducing measurement errors, and providing redundancy in the face of interference.

5.6 Overview

The preprocessing steps above lead to the preprocessed measurement vector with the following features:

$$\begin{aligned} \vec{M^*}_{ij} = [&T, Rssi_{init}, Rssi_{refl}, Rssi_{mean}, \\ &\vec{\phi_{offset}}, \vec{A_{init}}, \vec{A_{refl}}, \vec{A_{mean}}, \\ &\vec{SINR}_{init}, \vec{SINR}_{refl}, SINR_{sum}, \\ &ifft_{dist}, phase\_slope_{dist}, high\_prec_{dist}, Quality] \end{aligned} \tag{5.5}$$

Here, i refers to the initiator node, and j refers to the reflector node. Note that the last four elements are estimations made by a proprietary algorithm from Nordic Semi.

Chapter 6

Range Estimation

The input of the range estimator is vector $\vec{M}^*$, and the output is a vector $\hat{\vec{d}}_{ij}$, which contains two scalar values, a *distance* and *confidence*. The anchor node in the measurement is i in the range estimation, and j is the mobile/lock node. After the distance estimation, we no longer make a distinction between which node initiated the measurement, as the goal of the range estimator is to find a distance and confidence estimation based on $\vec{M}^*$, which is the preprocessed report of a single measurement, a take possible interaction like who was initiator into account in finding the distance.

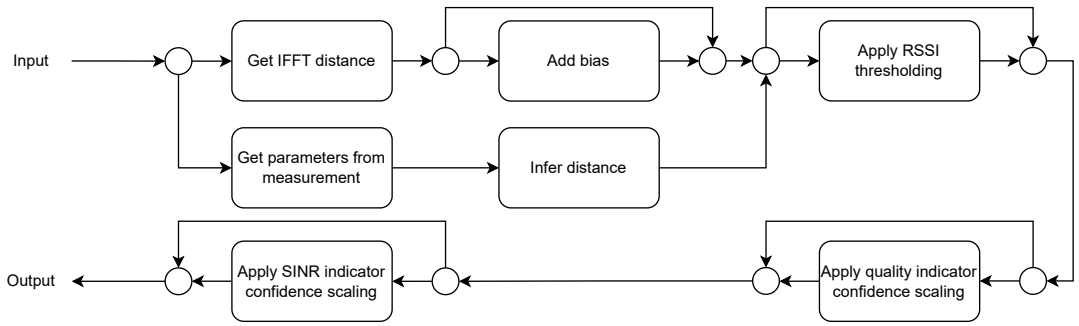


Figure 6.1: **Range estimator**

Figure 4.3 gives an overview of the range estimator. The circle indicates a junction where one of the outgoing paths can be chosen, which means that a certain function can be skipped or swapped for another function. The distance is by two methods: estimated by extracting the $ifft_{dist}$ from $\vec{M}^*$ and adding a bias or using a machine learning approach. The confidence is estimated using the $Rssi_{mean}$, $SINR_{sum}$ and $Quality$ features from $\vec{M}^*$. We will now explain these functions in detail.

6.1 Communication channel

When we want to estimate the distance between two devices, there are several metrics that we can use to find this distance. In chapter 2, we detail what principle can be employed. We try to estimate the distance using the wireless radio of a microcontroller.

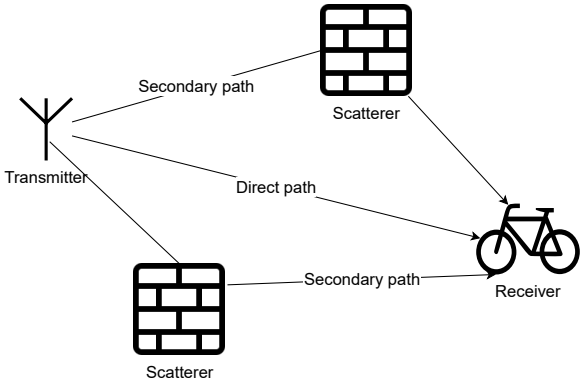


Figure 6.2: **Walls causing multipath fading**

6.1.1 Propagation channel distortions

A wireless communication channel is subject to interference from multipath fading and other sources in the same frequency band. Figure 6.2 visually displays the effect of multipath fading. In the example, the receiver has a direct line of sight with the transmitter, and due to path loss, the signal is attenuated when it arrives at the receiver. The signal also reflects off the environment (the house and tree), which is also received by the antenna. The length of this path depends on the reflecting object; hence, the signal has to travel farther than the line of sight case, causing interference. In a complex multipath environment, like a crowded bike storage environment, the interference undergoes this same principle. When the signal's path is blocked completely, we speak of a *no-line-of-sight* case, meaning there is no direct path in the communication channel. This is not necessarily a problem for communication, as data can still be transmitted, even if it is received via the wall.

6.2 Received signal strength distance estimation

The vector $\vec{M^*}$ contains measurement data that describes the channel behavior. We try to find the distance that describes this behavior best.

One feature from vector $\vec{M^*}$ is the RSSI, which can be modeled as follows:

$$RSSI_{rx}(dB) = P_{tx} + G_{tx} + G_{rx} + 20 \cdot \log_{10}\left(\frac{\lambda}{4\pi d}\right) \tag{6.1}$$

In equation 6.1, P_{tx} is the transmit power, G_{tx} and G_{rx} are the antenna gain at the transmitter and receiver respectively, λ is the wavelength and d is the distance. All variables are known except the distance, we can calculate the distance when we measure the RSSI. The model doesn't include antenna gain directionality; one would have to measure the radiation pattern and the angle between the devices and compensate for it to improve the distance estimation.

RSSI is the summation of all signals at the device's antenna, including multipath fading of the transmitter and other sources. This makes an estimation based on RSSI susceptible to noise from other interference sources. The multipath of the transmitter can be accounted for with an environment-specific compensation factor. This compensation factor must be determined using test data and is position-specific.

Kumar et al. [26] uses 802.15.4 hardware and achieves a mean error of 2.2 m in a controlled environment with line of sight and no interference from other sources. Our case would perform

worse due to the no line of sight conditions and interference, making only using the RSSI distance estimation approach not precise enough for the 1.5 m accuracy requirement.

6.3 Phase measurements

To understand how we use the phase measurement to find the distance, let us take a step back and consider radar systems [40].

6.3.1 Pulsed radar

In a pulsed radar system, a radio wave is sent in a certain direction, and we wait for the signal to reflect to the radar receiver. We can detect metallic objects like boats or aircraft since metal strongly reflects the signal. We can find the distance by measuring the delay between transmission and reception:

$$d = \frac{c \cdot t_{delay}}{2} \tag{6.2}$$

Here d is the distance, c is the speed of light, and t_{delay} is the round trip time of the signal.

6.3.2 Continuous wave radar

In continuous wave (CW) radar, we send an electromagnetic wave and listen for the reflections of the surroundings. The environment reflects the continuous wave. When an object moves, it causes a frequency shift due to the Doppler effect. By measuring this frequency shift, we can find the object's velocity by:

$$v = 2 \cdot \frac{f_d \cdot c}{f_0} \tag{6.3}$$

v is the velocity, f_d is the doppler frequency, c is the speed of light, and f_0 is frequency of the emitted wave. CW radar can only detect moving objects; stationary objects do not cause a frequency shift. The doppler frequency, f_d, due to the Doppler effect is very small; for a transmitter at 2.4 GHz (WiFi carrier frequency) and an object moving at the $300[\frac{m}{s}]$(speed of sound), this f_d is $\approx 4.8[KHz]$. This small shift can still be measured when the right processing is applied, like the superheterodyne receiver[50].

6.3.3 Frequency modulated continuous wave radar

In frequency-modulated continuous wave (FMCW) radar systems, we also change the frequency following a certain period pattern, like a sawtooth. In figure 6.3, we visualize the generated signal with the x-axis time and the y-axis frequency.

The green line displays the transmitted signal, and the red line displays the received signal. By measuring the shift in time between received and transmitted signals, we can find the distance between the reflecting object and the radar. Similarly, a shift in frequency between the wave currently transmitted and the wave received indicates the distance to an object. FMCW radar has the advantage over CW because it can measure both the speed with the Doppler effect and the distance using time markers (obtained from the frequency modulation), but CW lacks these time markers to find the distance. The distance can be found by using the following equation:

$$d[m] = \frac{c_0 \cdot |\Delta t|}{2} = \frac{c_0 \cdot |\Delta f|}{2 \cdot \frac{df}{dt}} \tag{6.4}$$

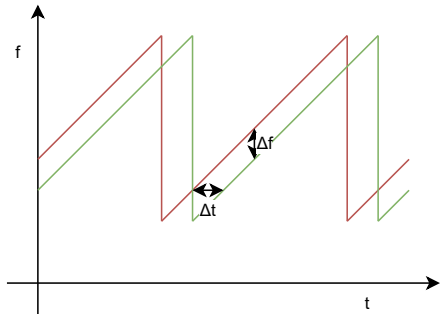


Figure 6.3: **FMCW radar, transmitted signal in red and received signal in green**

The distance between the transmitter and reflecting object is $d[m]$, $c_0[\frac{m}{s}]$ is the speed of light, $\Delta t[s]$ is the time delay, $\Delta f[Hz]$ is the frequency and $\frac{df}{dt}$ is the modulated shift in frequency per unit of time.

Let $f_1[Hz]$ and f_2 be the top and bottom frequency of the sawtooth, and $T[s]$ the duration of the pulse to go from frequency f_1 and f_2 linearly. The maximum distance that can be unambiguously distinguished then becomes:

$$d_{max}[m] = \frac{T}{2c} \tag{6.5}$$

The range resolution, the distance by which two targets can be distinguished, then becomes:

$$\Delta R[m] = \frac{c}{2BW} \tag{6.6}$$

Recently, there have been many advances in milliWave FMCW radar for applications in tracking cars [54], where a high granularity can be achieved in detected objects due to the high bandwidth.

6.3.4 Stepped frequency continuous wave radar

Similar to FMCW, stepped frequency continuous wave radar also modulates it's frequency In fig. 6.4, we see how stepped frequency continuous wave (SFCW) steps through using the step function in eq. (6.7).

$$f_n = f_0 + n\Delta f, \quad n = 0, 1, 2, \ldots, N \tag{6.7}$$

An advantage of SFCW over FMCW is that it uses discrete frequency steps that allow for simpler implementation.

6.3.5 Active reflector multi carrier phase difference

The multicarrier phase difference (MCPD) technique shares many similarities with the SFCW radar technology in finding the distance between targets. In MCPD, a pair of nodes find the phase difference over a range of frequencies, in our case, 2.400 GHz to 2.480 GHz with steps of 1 MHz, similar to the stepped frequency sweep of SFCW radar. In SFCW, the transmitted continuous is received at the radar after passively being reflected from the target. AR-MCPD discretely hops over frequencies (which could be in random order) and measures the amplitude and phase offset due to the propagation over the path for only one frequency at a time with the target's cooperation.

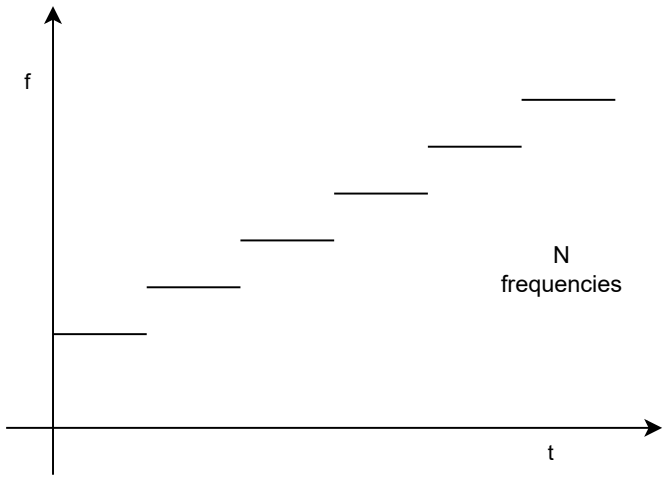


Figure 6.4: **SFCW radar**

The local oscillators are not synchronized; thus, we can not directly measure the phase offset at the reflector node. With the Active reflector principle, we can find the phase offset between the reflector and initiator node by using 2.3, and the propagation phase shift remains:

$$\begin{aligned}\theta_{offset} &= \theta_{init} + \theta_{refl} \\ &= (\theta_{lo_{init}} - \theta_{lo_{refl}} + \theta_{prop}) + (\theta_{lo_{refl}} - \theta_{lo_{init}} + \theta_{prop}) \\ &= 2 \cdot \theta_{prop}\end{aligned}$$

The combination of the narrowband signals resembles a wideband signal by measuring over a range of frequencies from 2.400 GHz to 2.480 GHz. We can reconstruct the channel impulse response in the time domain using the amplitude and phase measurements as in [6]. The channel impulse response is defined by:

$$h(t) = \sum_{i=0}^{L-1} \rho_i \cdot \delta(t - \tau_i) \tag{6.8}$$

The impulse response is $h(t)$, L is the total number of paths, ρ_i is the amplitude of path i, δ is Dirac delta function, and τ_i is the delay of path i. The delay τ_i corresponds to the signal's travel time through the channel. By multiplying this delay with the propagation speed of the signal, approximately c for 2.4 GHz in air, we can find the distance the signal traveled by:

$$d = \tau \cdot c \tag{6.9}$$

In fig. 6.5, we visualize the channel impulse response.

Multiple paths exist between the transmitter and receiver, and the different copies arrive at the receiver at different times. The different peaks of τ_i represent the signal's different paths through the channel.

6.4 IFFT distance estimation

The goal of the **ifft** based distance estimation is to use the set of phase measurements from 2.400 GHz to 2.480 GHz to find the distance. In section 6.3.5, we explain that the amplitude and phase meas-

urement represents the channel's impulse response. Consider the discrete Fourier transform in eq. (6.10).

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi kn/N} \tag{6.10}$$

$X[k]$ is the complex frequency component at frequency bin k, $x[n]$ is the discrete time signal, N is the number of samples and n the sample index. In section 6.3.5, we used the MCPD to find the impulse response, and we can employ the DFT to find the impulse response in the time domain.

If we use $\vec{A_{mean}}$ and $\vec{\phi_{offset}}$ from vector 5.5, we can use eq. (6.11) to find a discrete Fourier series.

$$X[k] = A_{mean_k} \cdot e^{j\angle\phi_{offset_k}} \tag{6.11}$$

In eq. (6.8), we said the channel impulse response is a summation of different paths with delay τ_i. If we assume $X[k]$ is the $H[k]$, we can use the discrete inverse Fourier transform in eq. (6.12) to find $h[n]$, which is the sampled $h(t)$ of eq. (6.8).

$$h[n] = \frac{1}{N} \sum_{k=0}^{N-1} H[k] \cdot e^{j2\pi kn/N} \tag{6.12}$$

We can then find the path delay of the $n-th$ sample using eq. (6.13) [57].

$$\tau_n = \frac{h_n}{N \cdot f_s} \tag{6.13}$$

Where N is the total number of samples(80 in our case), and F_s is the sampling frequency (1 MHz in our case). This means the distance between the bins becomes 3.75 m using 6.9. We can now employ a peak-finding algorithm to find the strongest paths, which we will demonstrate in chapter 9. Due to multipath effects, the strongest path doesn't necessarily have to be the shortest (see 6.5). Hence, a smart peak selection algorithm has to be employed to find the short path distance.

Due to the limited bandwidth of 80 MHz in MCPD 6.3.5, we can not separate paths that differ by less than 3.75 m. This is similar to the range resolution of FMCW radar in 6.3.3 using 6.6, except that we omit the 2 since we measure a one-way trip. We can, however, zero pad $X[n]$ to get more granular time steps in our delay profile, so if the paths are properly separated, we can obtain smaller steps between peaks. The resolution is now limited to the error of our IQ measurements [45].

The value $ifft_{dist}$ from vector 5.5 uses a similar method to obtain distance. We will discuss the performance of $ifft_{dist}$ in chapter 9.

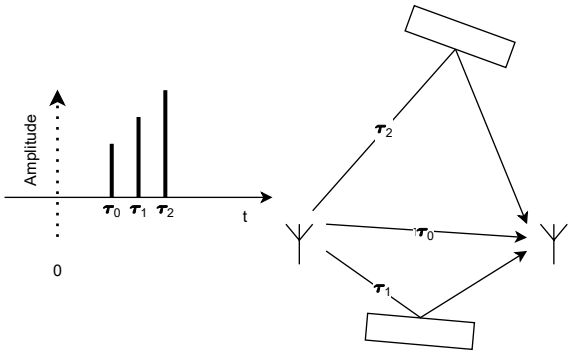


Figure 6.5: **Channel impulse response**

6.5 Machine learning distance estimation

The measurements in section 4.2 are subject to complex multipath fading depending on environmental factors and device-specific effects like oscillator offset. In fig. 6.6, an overview of machine learning approaches is shown; Bonaccorso [7] gives an extended overview of machine learning algorithms. Conventional methods using the RSSI 6.2 and IFFT section 6.4 can provide results with an accuracy of within one meter in line of sight conditions [45], [52], but the harsh no line of sight condition present in bike storages are not evaluated.

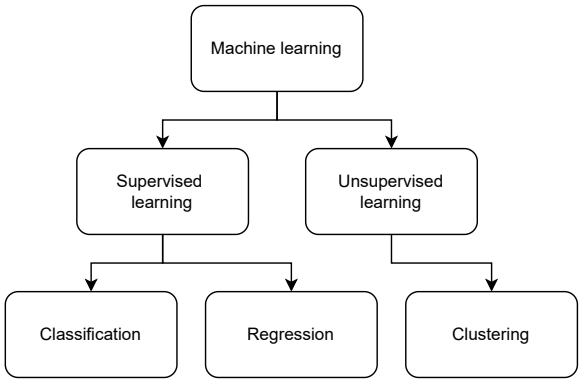


Figure 6.6: **Machine learning overview**

6.5.1 Challenges addressed by Machine learning

The input for the position estimation algorithm in chapter 7 is distance matrix **D** from eq. (4.1) consisting of vectors $\hat{\vec{d}}_{ij}$, where $\hat{\vec{d}}_{ij}$ is the distance between node pair i and j with a confidence between 0 and 1. If the distance is found using, for example, the ifft method in section 6.4, and there is no direct path between receiver and transmitter, the distance will be an overestimation of the actual distance. If we then employ a naive trilateration method to find the position, the node will be estimated to be farther away from the anchor nodes than the physical direct path [21].

The no line-of-sight NLOS cases must be accounted for in the full algorithm, so we either compensate directly in distance estimation or the positioning algorithm. It is important to separate the NLOS measurement from the LOS measurement. However, this is hard since an LOS measurement with many multipath causes similar communication channel distortions as NLOS measurements, but this is usually ignored [43].

Nesse et al. [31] state two main approaches to the NLOS problem: first, identify the NLOS and LOS condition and then compensate or compensate directly. We directly estimated the distance based on preprocessed measurement vector$\vec{M^*}_{ij}$, since the most simple approach.

6.5.2 Objective and evaluation metrics

The objective of our machine learning algorithm is to find the distance that most closely represents the direct distance between a node pair. This means that even if a wall separates two nodes, it should estimate the distance of the direct path even in no-line-of-sight conditions.

When our machine learning model estimates the distance, it makes a certain scalar distance prediction, which has an error (positive or negative) with the reference distance between node pairs.

We evaluate our algorithm by how much this error is over a range of measurements We split our data set into a train and test set; how we get our test and train set is explained in section 6.5.6.

We evaluate the performance of our algorithm by the Mean Absolute Error MAE given in eq. (6.14):

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{6.14}$$

In eq. (4.1), N is the total number of measurements, i is the index of the measurement, y_i is the reference distance of measurement, and $\hat{y}_i$ is the predicted (or inferred) distance. We choose the MAE as it is an initiative metric since the unit is in meters, just like the distance estimate itself, and it is used by many reference works, so we can compare performance easily [45], [61].

6.5.3 Feature selection

The preprocessed measurement vectors $\vec{M^*}_{ij}$ from eq. (5.5) are processed as a separate entity, meaning we do not consider related measurements like averages over multi-repetitions. This approach was chosen as it is the most flexible and simple. A distance estimate can be made with a single measurement, and we do need to extract features from multiple measurements and find statics. We can repeat a measurement multiple times to improve accuracy, but this is handled by the position estimation algorithm discussed in chapter 7.

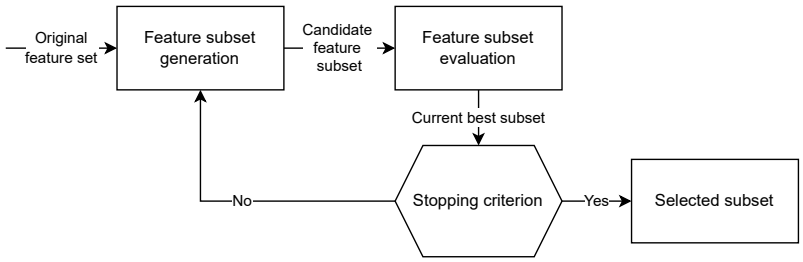


Figure 6.7: **Feature selection procedure**

From the available data in measurement vector $\vec{M^*}_{ij}$, we try to select a set of features that make our algorithm perform best. We used forward sequential feature selection (FSFS) to find the best set of features. FSFS is visualized in fig. 6.7; you start with zero features and test the model's performance with all features individually, and then we select the best-performing feature. Next, redo the process for the remaining features until the stopping criteria are met.

In chapter 9, we will show which features were selected for the final model.

6.5.4 Model selection

Machine learning models can be divided into classifying and regression models [52]. Classifiers try to predict which "class" a certain input belongs discretely, and regressors predict a continuous value. Cortise et al. [12] use wkNN to estimate the position of a wireless node and combine the RSSI and IFFT features to find the position. Their approach does not scale well, as many measurements are required when applied in a large area.

They claim that wKNN is good for implementing on an embedded device as it only requires training data for every measurement, but this does not scale well in large areas where a lot of training data with many features is required. All these stored data could contain redundant information that does not benefit the algorithm's performance.

To overcome this limitation, we train a Multi-layer perception network shown in fig. 6.8.

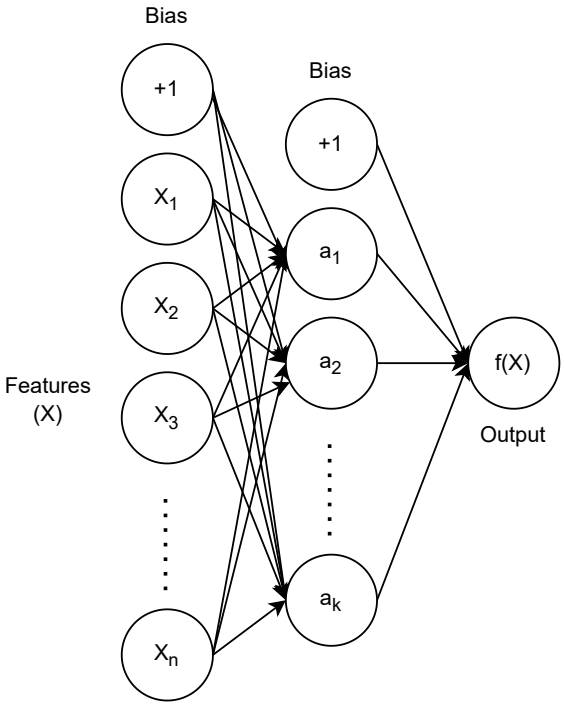


Figure 6.8: **Multilayer perception neural network**

Multilayer perception neural networks have the advantage that they can learn non-linear relations and be trained online, meaning that an already trained model can be trained further without reinitialization.

Having a model that can run on the bike and anchor node (see fig. 3.1), the network traffic on the low-power network is minimized for exchanging training data. The initial model can be trained offline on a powerful machine without constraints on the training set. Embedded devices can nowadays contain a memory of up 1 MB, which allows us still to create a model of up to 250000 weight using a 32-bit floating point number. The final model will contain far fewer weights, as shown in chapter 9.

6.5.5 Hyperparameter tuning

Hyperparameters are the model settings not learned from the training data. These settings include the number of hidden layers, neurons per layer, activation functions, etc. The selection of these parameters can greatly impact the model's performance, and systematic optimization is required for optimal performance. We tune our parameters in the following fashion following the suggestions from [60]:

- Learning rate
- Momentum
- Mini-batch size
- Hidden layers

- Learning Rate decay
- Regularization
- Activation function

In chapter 9, we show the final optimal parameters found.

6.5.6 Train and test data

We verify our trained machine-learning model by providing a test set and using the conventional machine-learning method as a reference. We split our total set of measurements into a training and test set. Every distance matrix D measurement was made eq. (4.1) under the same spatial conditions. To prevent overfitting, we put all measurements of the distance matrix in either the training or test set.

In fig. 6.9, we see how cross-validation by changing the test and train set works.

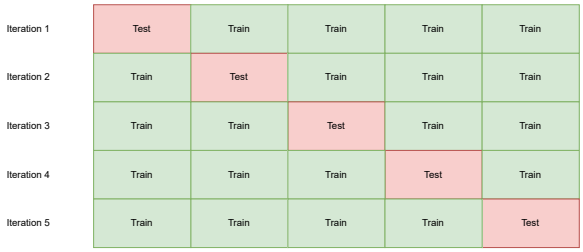


Figure 6.9: **K-fold cross-validation scheme**

The training set is divided 80 : 20, and the test and train sets are interchanged. We use this approach and randomly shuffle the set of distance matrices D, and also 80 : 20. The ratio 80 : 20, as it has the advantage of having a relatively big train set for getting the optimal model, and the test set is not too small that many repetitions are needed for the mean performance to stabilize.

6.6 Confidence estimation

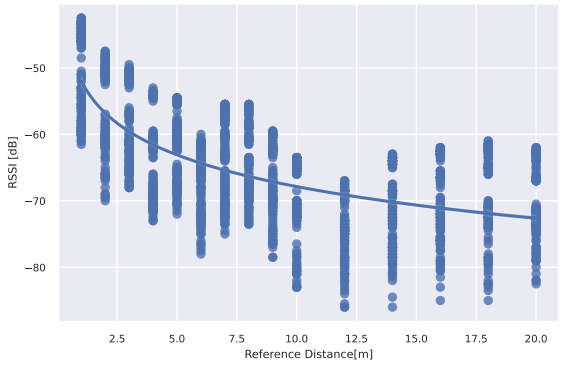
Every distance estimation is based on a preprocessed measurement report eq. (5.5). However, depending on the channel conditions between the measured node pair, we get a certain noise level due to multipath effects, interference, and no-line of sight conditions.

Measurement vector $\vec{M^*}_{ij}$ offers several parameters we can use to estimate the accuracy of the predicted distance.

6.6.1 Received signal strength indicator thresholding

Most wireless communication devices expose the Received Signal Strength Indicator (RSSI) when receiving a message. Due to the nature of RSSI, it is only suitable for a broad estimation for finding the distance between sender and receiver.

The RSSI indicates the energy level measured at the antenna at a specific frequency, including noise due to interference and multipath fading, making it hard to separate the signal of interest. In section 6.6.1, we see a plot of RSSI versus reference distance, showing the large spread in measured RSSI for a certain distance.



Despite this, distance estimation levels of MAE below 3[m] are possible if many repetitions are measured [26].

We utilize the broad estimation of RSSI to filter faulty measurements using eq. (6.15):

$$RSSI\_Threshold_{conf}(\vec{M^*}_{ij}, d) = \begin{cases} 1 & \text{if } RSSI_{avg} \geq T_{Rssi}(d) \\ 0 & \text{if } RSSI_{avg} < T_{Rssi}(d) \end{cases} \tag{6.15}$$

Where $\vec{M^*}_{ij}$ is the measurement vector, $d_{estimated}$ is the specific distance, $T_{Rssi}(d)$ represents the threshold function that depends on the distance $d_{estimated}$. The function $T_{Rssi}(d)$ was found empirically and implemented using a lookup table.

6.6.2 Quality indicator

The preprocessed measurement report $\vec{M^*}_{ij}$ gives a *Quality* indicator[1], which we can utilize to assign a confidence level aswell. The confidence function based on the quality indicator is defined as:

$$Quality\_Indicator\_conf(\vec{M^*}_{ij}) = \begin{cases} 1 & \text{if } Quality = \text{"}good\text{"} \\ C & \text{if } Quality = \text{"}bad\text{"} \end{cases} \tag{6.16}$$

Where *Quality* is the quality indicator, which can be either "good" or "bad", c is a value between 0 and 1, representing the confidence level associated with a "bad" quality indicator. The value of c can be adjusted to reflect the desired confidence level for "bad" quality.

6.6.3 SINR

The preprocessed measurement report $\vec{M^*}_{ij}$ gives a Signal-to-noise-plus-interference indicator per measured tone (2.400 GHz to 2.480 GHz, with steps of 1 MHz). A higher value SINR value indicates more noise, and vice versa.

Let $\vec{V}$ be the vector containing signal-to-noise-plus-interference indicators, where each indicator can take values 0, 1, or 2. The sum of the vector is denoted as $S = \sum_{i=1}^{n} V_i$, where n is the length of the vector. The confidence assignment function is defined as:

$$SINR_{\text{conf}}(\vec{M^*}_{ij}) = \begin{cases} 0 & \text{if } SINR_{sum} > SINR_{max} \\ \frac{SINR_{max}}{SINR_{sum}} & \text{if } SINR_{sum} \leq SINR_{max} \end{cases} \tag{6.17}$$

[1]The algorithm to find the Quality Indictor is Closed Source

In eq. (6.17) $SINR_{sum}$ is the sum of $SINR$ values of all tones, $SINR_{max}$ is the maximum allowed sum value.

The $SINR_{max}$ can be adjusted to fit certain requirements; a lower value will discard more measurements, which means more measurements are assigned zero confidence.

6.7 Overview

The distance estimation steps above lead to the following range estimation output:

$$\vec{R_{ij}} = [\hat{d}, \hat{c}] \tag{6.18}$$

Where $\vec{R_{ij}}$ is the range estimation between anchor node i and bike node j, $\hat{d}$ is the distance estimation, and $\hat{c}$ is the confidence estimation.

Chapter 7

Position Estimation

An overview of the position estimator is given in fig. 7.1, the input is a series of distance estimations $\vec{R_{ij}}$ from eq. (6.18) and the output is a position $\vec{pos} : (x, y)$ and a classification **Correctly Parked** or **Not Correctly parked**.

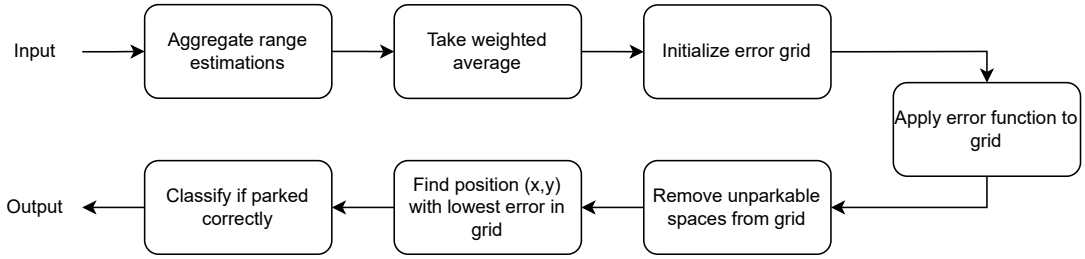


Figure 7.1: **position estimation**

First, a series of range estimations for a position request are aggregated. Every measurement consists N of repetitions. We take the weighted average of the N range estimation repetitions for every anchor node. Then, after zero initializing an error grid, we apply the error to all points in the error grid. Next, we remove the spaces where the bike can not be parked. The minimum value in the error grid is selected as the most likely position. Finally, we classify whether that position is allowed parking space and return the position and classification to the user.

7.1 Aggregate estimations

Let k be the number of anchors in a bike storage. We define the set of anchors as follows:

$$\boldsymbol{\mathcal{A}} = \{1, 2, 3, \ldots, k\}$$

In this notation, $\boldsymbol{\mathcal{A}}$ represents the set of anchors from 1 to k.

When localization for a bike is requested, we schedule measurements according to the description in chapter 4, and we start to fill matrix in eq. (4.1) with measurements. Measurements must be scheduled sequentially. They are directly transmitted to the base station after the measurement has finished, so they also arrive sequentially.

The measurements are aggregated per anchor at the range estimator aggregator as follows:

$$\mathbf{d}_k = \begin{bmatrix} \hat{R_{r_1}} \\ \hat{R_{r_2}} \\ \vdots \\ \hat{R_{r_n}} \end{bmatrix} \tag{7.1}$$

In eq. (7.1), k is the anchor index, and r_n is the $n-th$ repetition of range estimation between the target bike and anchor k.

7.1.1 Take weighted average

Our position estimator uses a single-distance confidence pair for all N repetitions. The confidence level of the measurement indicates how sure we are of our distance estimation. Hence, we want the final distance estimation to consider the confidence level if we take multiple measurements. We employ a weighted average, see eq. (7.2) for this.

$$\hat{d_{est_k}} = \frac{\sum_{i=1}^{N}(\hat{d}_i \cdot \hat{c}_i)}{\sum_{i=1}^{N} \hat{c}_i} \tag{7.2}$$

Where d_{est_k} is the distance estimation of the target bike and anchor k, $\hat{d}_i$ represents the distance estimate of repetition i, $\hat{c}_i$ represents the confidence estimation of repetition i and n is the total number of repetitions. The confidence estimate is found by taking the mean of all confidences:

$$\hat{c_{est_k}} = \frac{\sum_{i=1}^{N}(\hat{c}_i)}{N} \tag{7.3}$$

This leaves us with the following distance and confidence vectors $\vec{dist}_{est}$ and $\vec{conf}_{est}$ for anchors 1 to k:

$$\vec{dist}_{est} = \begin{bmatrix} \hat{d_{est_1}} \\ \hat{d_{est_2}} \\ \vdots \\ \hat{d_{est_k}} \end{bmatrix} \tag{7.4}$$

$$\vec{conf}_{est} = \begin{bmatrix} \hat{c_{est_1}} \\ \hat{c_{est_2}} \\ \vdots \\ \hat{c_{est_k}} \end{bmatrix} \tag{7.5}$$

7.2 Position estimator

After obtaining the final distance-confidence pairs for all k anchor nodes, we do a brute-force grid search to find the position of the target bike node. The size of the grid is chosen to be 20 m outside the boundaries allowed parking area of the bike storage, as defined in section 3.1.

7.2.1 Error function

We define the error function based on the work of [2]. First, we define the evaluation and anchor node positions as $\vec{r_{eval}} = [x, y, z]$ and $\vec{r_{anchor_i}} = [x, y, z]$.

Let k be the number of anchors in a bike storage. We define the set of anchors as follows:

$$\boldsymbol{\mathcal{A}} = \{1, 2, 3, \ldots, k\}$$

In this notation, $\boldsymbol{\mathcal{A}}$ represents the set of anchors from 1 to k. The error function then becomes 7.6:

$$Error(\vec{r}_{\text{eval}}) = \sum_{i=1}^{k} (conf_{est_i} \cdot ||\vec{r_{eval}} - \vec{r_{anchor_i}}| - \hat{d}_i) \tag{7.6}$$

The position can then be found by finding a vector $\vec{r}_{\text{eval}}$, that minimizes the error function in eq. (7.6).

7.2.2 Grid search

The error function in eq. (7.6) is non-convex, meaning there can be multiple local minima [18]. The resolution required by our algorithm is limited; hence, we employ a brute-force grid search to find the global minimum (inside our grid). We loop over all possible positions inside our grid with predefined step size and record the error for every step in a matrix.

7.2.3 Matrix representation

Consider a matrix $\mathbf{S}$ representing positional errors, where the rows correspond to y coordinates, and the columns correspond to x coordinates. Each element $M_{i,j}$ in the matrix represents the error at the coordinate (x_j, y_i).

Let n be the number of rows, m be the number of columns, Δy be the predefined step size for y coordinates, and Δx be the predefined step size for x coordinates.

The y coordinates for row i are given by:

$$y_i = i \cdot \Delta y, \text{ where } i = 0, 1, 2, \ldots, n-1$$

The x coordinates for column j are given by:

$$x_j = j \cdot \Delta x, \text{ where } j = 0, 1, 2, \ldots, m-1$$

With this setup, you have a matrix $\mathbf{S}$ where each element $\mathbf{S_{i,j}}$ represents the error associated with the coordinate (x_j, y_i) in meters.

We define the vector of distance estimations as follows:
In this notation, $\vec{dist_{est}}$ is the vector of distance estimations from anchor 1 to k.

We define the vector of confidence estimations as follows:

$$\vec{conf_{est}} = \begin{bmatrix} \hat{c_1} \\ \hat{c_2} \\ \vdots \\ \hat{c_k} \end{bmatrix} \tag{7.7}$$

In this notation, $\vec{conf}_{est}$ is the vector of distance estimations from anchor 1 to k.

The position vector $\mathbf{r}$ with elements x, y, and z is defined as follows:

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{7.8}$$

In this notation:

x represents the x-coordinate of the position.
y represents the y-coordinate of the position.
z represents the z-coordinate of the position.

We define $\vec{loc}_{est}$ as the estimation

$$\vec{conf}_{est} = \begin{bmatrix} \hat{c_1} \\ \hat{c_2} \\ \vdots \\ \hat{c_k} \end{bmatrix} \tag{7.9}$$

In this notation, $\vec{conf}_{est}$ is the vector of distance estimations from anchor 1 to k.

7.2.4 Remove unparkable spaces

In the initialization step (see 3.1), we define a set of polygons in which the bike can not be physically parked. These areas are, for example, private buildings of people living next to the train station. For all elements in $\mathbf{S}$, we check if they are in the polygons where the bike can not be parked, and if that is the case, we set the error to zero.

7.3 Find position and classify park status

The last step is to find the minimum value of $\mathbf{S}$ and find the corresponding position. Then, we check where this is in the *Allowed* parking area and return the result to the user.

Chapter 8

System Implementation

This chapter gives an overview of the system architecture of *SkopeiLocate*. The goal of *SkopeiLocate* is to provide the position of bikes in a bike rental system. This position is then used to determine if a user has correctly parked a bike for terminating a rental. The location also informs the next user where to find the bike, enabling operators to find their bikes and perform maintenance.

8.1 Requirements

The first step in solving the problem bike position problem, as stated in chapter 1, is to define the system requirements. These requirements constrain our design space to a subspace of proposed approaches in chapter 2 of existing work. The system needs to be integrated with the existing infrastructure of the Skopei bike rental system, and the requirements result from this condition. The requirements are as follows:

Integrate with the deployed system

The locks and routers use specific hardware and software packages, and *SkopeiLocate* should be integrated with the existing infrastructure.

Determine the location of the bike

The system must find the bike's location within the bike storage and determine if it is within or outside the allowed parking area. A bike has a size of $\approx 1.5\,\text{m}$, which we set as the accuracy requirement within the valid parking zone. Millimeter precision does not make sense, as a bike can still easily be located if it is in a radius of a few meters. High accuracy enables operators to form stricter parking zones. Due to the size of the bike, an accuracy of more than $1.5\,\text{m}$ does not benefit the system significantly.

Operate in various bike storage environments

Skopei's bike rental system is currently deployed in multiple bike storage environments with different characteristics regarding walls, metallic obstructions, and floor plan layouts.

Scale with an arbitrary number of bikes and routers

Some bike storages only contain two routers and ten bikes, while the large locations go up to thirty routers and a thousand bikes. The system should operate either of these extremes.

Operate in real-time

When the bike is locked at the bike storage, the system needs to check within a limited time whether the bike is parked correctly and notify the user. The latency of the existing system takes $\approx 2\,\mathrm{s}$ to complete. We aim to reach the total positioning time within this time frame, creating a maximum added overhead of $2\,\mathrm{s}$.

Cause minimal overhead

It is also crucial not to add disruptive overhead to network communications, battery consumption, and memory usage of the smart lock. Battery power and network overhead should not rise by more than $20\,\%$

8.2 System infrastructure overview

The full solution consists of a network of Routers with known fixed locations (anchor nodes), bike locks (mobile nodes) with unknown locations, and a base station for processing. An overview of the system is illustrated in fig. 8.1.

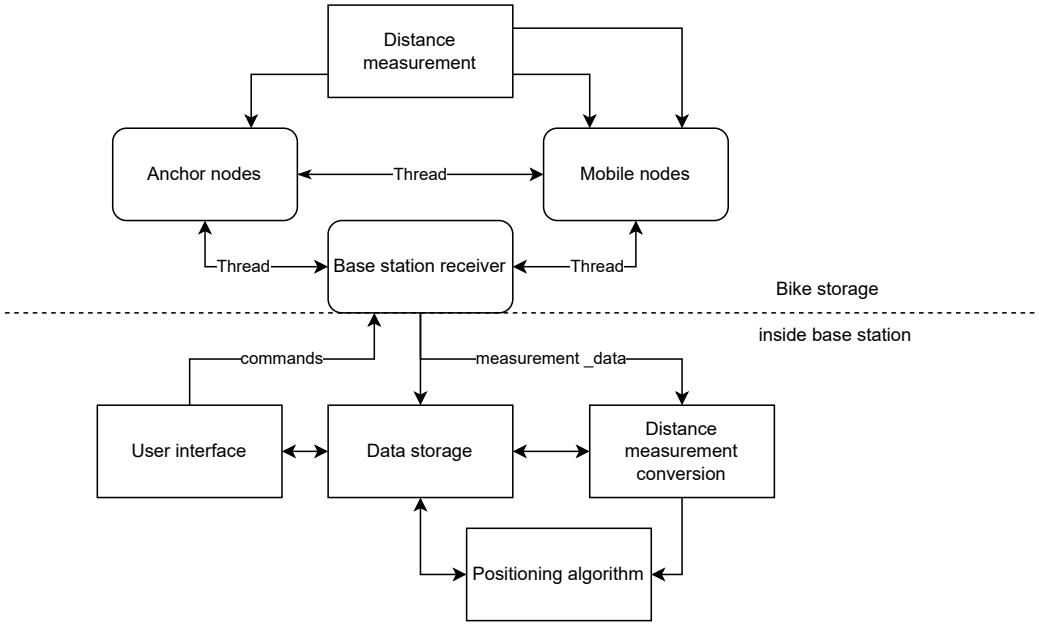


Figure 8.1: **System overview**

The bike lock and routers are powered by a *Nrf52840* [35] microcontroller, and the base station is a generic laptop running *Ubuntu 22 LTS*. The code running on the base station was implemented using Python, as it comes with good community support and many freely available packages like

Numpy, Pandas, and Sklearn. The NrfConnectSDK, used for the firmware on the routers and bike nodes, is written in C/C++ and is based on the Zephyr-rtos.

The base station wirelessly communicates with node pairs to send commands, initiate distance measurements, and receive raw data. *SkopeiLocate* implements the following steps to answer a localization request:

1. Select a target bike node
2. Schedule distance measurements with all k anchor node pairs and do N repetitions
3. Store measurement reports
4. Find the position using the positioning algorithm using the range estimations
5. Return position

The bike-sharing solution of Skopei can then use this position to determine if user are allowed to terminate their rental and other position-related services.

8.3 Communication infrastructure

Skopei uses Thread for communication with the bikes [22]. This communication network can be reused to transmit data between the base station, mobile, and anchor nodes. The Thread USB dongle adds a thread network interface to the base station. The system can now communicate with the nodes in the Thread network using network sockets.

The UDP protocol was chosen to transmit the packages as it is simple, adds little network overhead, and is the preferred method by Thread. It is connectionless, meaning no connection needs to be set through a handshaking protocol, which allows for low message latency. Unlike TCP, package loss needs to be handled by handling retransmission manually.

After attaching to the network, the nodes periodically send a heartbeat message containing the nodes' ID, node type, battery level, and local time to the base station via UDP. Using these heartbeats, the base station keeps track of the list of active nodes. The base station can send the following commands over UDP to the active nodes:

- `Beep` - Activate device buzzer for 200 milliseconds
- `Schedule_dm` - Schedule a distance measurement at a specified time
- `Report Resend` - Request the last report to be resend
- `Set offset` - Set offset of the local clock of a Device for time synchronization
- `Ping` - Request current local time from device

The beep command can verify which node ID is linked to which physical device. A measurement can be started by sending a schedule_dm command with a starting time to a device pair. After the measurement, both devices transmit the result to the base station. In section 8.4, we expand on how distance measurements are taken.

All communication is over UDP, which does not guarantee the arrival of packets. The larger the packet size, the more often packets contain bit errors and are dropped. If the report is not received after the measurement, we can request a retransmission of the latest report. We could schedule a measurement by indicating a wait time, and if we instruct both devices at the same time to wait a specific time, they will both start the measurement at the same time. This requires that the

command is received at the same time by both devices. We found that the transmission time of a message can vary by more than 10 ms depending on network activity, and the problem worsens for every added hop.

We resolve this problem by synchronizing the local time on all devices with the Unix time of the base station. The base station then instructs nodes to start a measurement at a specific Unix timestamp. The local time of a device consists of the on-time of the device added to an offset set with the set_offset command. We find the offset by sending multiple ping commands to the device to find the average round-trip time to the device. Using the local time the device returns on every ping, we can find the offset between the base station and the device to synchronize them within 1 ms. Due to oscillator imperfections, the clock between devices and the base station can drift by 10 ms over several minutes. Hence, we need to resynchronize periodically.

8.4 Distance measurement

The NRFConnectSDK provides a closed-source distance measurement toolbox that provides distance measurements between two Nrf52840 devices. The firmware of the bike locks already uses the NRFConnectSDK, enabling easy integration with the existing firmware. During a measurement, one device is the reflector, and the other is the initiator. The `schedule_dm` command looks as follows:

$$\texttt{schedule\_dm}(Timestamp, ID, Role, HoppingSequenceSeed)$$

The *timestamp* indicates at which time the measurement should be started. The reflector must start listing before the initiator starts the measurement [34]. The waiting period for the reflector can be up to 20 ms, allowing for more lenient clock synchronization between devices. Better synchronization means less time is wasted on waiting, allowing for more measurements in the same period. Also, we found that measurements are more likely to fail for larger clock offsets, and all measurements fail for an offset of more than 20 ms. *ID* is used to differentiate between measurements, and the *role* tells the device if it should act as an initiator or reflector. The measurement uses a technique called multicarrier phase difference (MCPD) [38] with the following steps [34].

1. The initiator transmits a constant tone at a certain carrier frequency
2. The reflector compares the received tone with its local oscillator to find the phase offset
3. The reflector transmits a constant tone at the same carrier frequency.
4. The initiator compares the received tone with its local oscillator to find the phase offset
5. Repeat for all tones
6. Calculate the distance between the devices based on the phase offsets

The *hopping sequence seed* generates the random sequence of tones that the reflector and initiator hop over. Every measurement uses a randomly generated seed, which allows us to test for robustness against different hopping sequences.

A measurement provides the following data:

- Raw Data
  - Timestamp
  - Status Code

  - Remote and local RSSI
  - Local and remote IQ measurements for per tone
  - Local and remote SINR indicator per tone
  - Quality indicator 0(good), 1(medium) or 2(poor)
- Distance estimations
  - Pathloss distance estimation
  - IFFT-based distance estimation
  - Phase-slope distance estimation
  - High-precision distance estimation

The measurement data consists of raw data and processed distance estimates. The processed distance estimates are calculated using the raw data.

The timestamp is the time after synchronization with the base station when the measurement was scheduled. The status says whether the measurement was successful or not. The received signal strength indicator (RSSI) for the remote and local devices is given. The values are flipped depending on whether the report originates from the perspective of the initiator or the reflector.

The phase offset between the devices is found using Inphase and Quadrature (IQ) measurements. We can find the phase offset to the device's local oscillator by taking the argument of the I and Q components. Eq. 2.6 is then used to find the phase offset due to the signal propagation. The tones are between 2.400 GHz and 2.480 GHz with steps of 1 MHz. The phase offsets of all tones form the channel spectrum. We also have a signal-to-interference plus noise ratio indicator per tone for the reflector and initiator. The value can be 0,1 or 2.

The exact steps to find the processed distance estimates are unknown as the library is closed-source. However, Nordic indicates [33] that the path loss estimate is based on eq. (2.1), the phase slope is based on eq. (2.5), and the FFT uses the Inverse Fast Fourier Transform (IFFT) over the channel spectrum. The high-precision distance algorithm uses undisclosed advanced spectrum techniques but requires long computation time and high memory.

Chapter 9

Evaluation

This chapter evaluates the proposed system, comparing its performance and viability through comprehensive tests. The goal is to provide insight into the system's behavior in real-world scenarios, highlighting its strengths and identifying potential areas for improvement. It is structured as follows: First, we define the definition for ranging and position estimation performance, then show the experimental setup and give an overview of the evaluated measurement scenarios. Next, we present the raw measurements, then we evaluate our range and position estimation.

9.1 Definition of ranging and positioning performance

Before evaluating, we first define the performance metrics for the range and position estimators.

Our system can broadly be subdivided into three steps: measurement, distance and confidence estimation, and position estimation. We evaluate each step separately, this approach keeps evaluation simpler.

We make a distinction between accuracy and precision. An accurate system performs well on average if many estimates are made, while a precise system has little spread between the estimations but, on average, has an error. Ideally, we want both an accurate and precise system; this way, with minimal measurements, our error is small.

We use the Mean Absolute Error (MAE) for the error. It is calculated as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where:

$$\begin{aligned} n &: \text{Number of data points} \\ y_i &: \text{Actual value for data point } i \\ \hat{y}_i &: \text{Predicted value for data point } i \end{aligned}$$

Furthermore, we use visualizations of the errors, like the boxplots or the cumulative distribution function, to get insight into precision and accuracy. These visualization give more insight into the behavior and origin of the error.

Figure 9.1: **Measurement setup**

9.2 Experimental setup

Figure 9.1 shows all hardware used for the measurements. It contains the following items:

1. Bosch PLR30C Digital Laser Measure [14]
2. BMD340 Stand-alone Bluetooth 5 low energy and IEEE 802.15.4 module [53]
3. nRF52840 Dongle [35]
4. 1 Skopei Bike lock
5. 8 Anchor Devices
6. Lenovo T14 laptop with Ubuntu 22.04.3 LTS

The Bosch PLR30C digital laser measurement device, which can measure up to $30 \pm 0.002[m]$, was used to find the dimensions of the test area and get the reference position of the Anchor Nodes and Lock for every measurement. The PLR30c distances to walls can be determined within a few seconds to make measurements easy and quick while meeting the precision requirement of less than 1.5 m given in section 8.1.

The nRF52840 Dongle is plugged into the laptop, adding the Thread network to the available network interfaces. Other devices on the Thread network can be reached via sockets, similar to other computers on a Wifi network.

The BMD340 is a standalone System on Module with an nRF52840 microcontroller suitable for a low-power embedded system and is compatible with the Zephyr-based nRF Connect SDK. It is available in the Anchor and Lock devices, is used to connect them to the Thread network, and is supported by the Nordic Distance Measurement library [46].

Finally, we have 8 Anchor nodes and one Lock node to do the measurement. The Anchor nodes agree with the operation requirements from Skopei for their bike-sharing solution. The lock node was attached to a bike during the experiment, and the anchor nodes were mounted on the wall with double-sided tape.

9.3 Measurement scenarios overview

We gathered distance measurements in two scenarios: An *open field* and a *bike storage environment.* The open-field scenario was chosen to minimize environmental factors like reflection off walls and interference with other $2.4GHz$ transmitters. As the final system will be deployed in bike storage, we gathered measurement data in bike storage that represents an average bike storage environment.

9.3.1 Line of sight baseline

The baseline test was performed to evaluate the system's performance in an environment with minimal interference from surrounding objects like walls and bikes. The measurement setup is shown in fig. 9.2. We chose plastic and wood poles as they are expected to have lower interference than metallic ones.

Lock node on plastic pole at 0 degrees

Anchor nodes on wood pole with 90 degrees offset

Figure 9.2: **Open field measurement setup, measurement rope indicates the direction of the opposing node**

The orientation of the lock node was varied by rotating it clockwise with steps of 90 degrees to get a total of 4 different orientations for each measurement. The two anchor nodes (Kilo and Juliett) were placed at an offset of 90 degrees. Anchor Juliett has its antenna facing toward the lock and anchor Kilo facing away. These different orientations allow us to find the effect of different angles while the distance stays constant. The distance is varied with steps of 1 meter for the range 0 through 10 meters and with steps of 2 meters, between 10 and 20 meters. Every measurement was repeated 10 times, and the total amount of measurement then becomes:

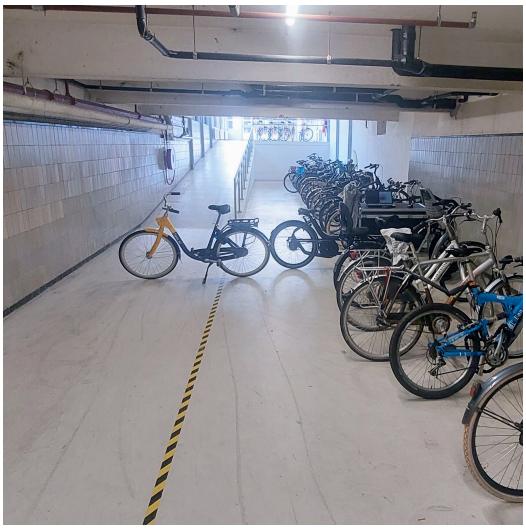
$$\begin{aligned} N_{measurements} &= N_{repetitions} \cdot N_{angles} \cdot N_{distances} \cdot N_{roles} \cdot N_{anchors} \\ 2400 &= 10 \cdot 4 \cdot 15 \cdot 2 \cdot 2 \end{aligned} \tag{9.1}$$

Out of these 2400 measurements, we filtered out 203 failed measurements. A measurement is deemed failed when the status code is -1 or when the measurement values between the reflector and initiator do not match.

9.3.2 Bike storage

We use the bike storage environment to test our system in a realistic environment, representing the conditions in other bike storage environments. We show an overview of the measurement setup in fig. 9.3. On the left, we see the bike (with the bike lock node) used for testing; on the right, we see

Lock node attached to bike



Overview bike storage test environment

Figure 9.3: **Bike storage measurement setup**

a picture from within the bike storage. In fig. 9.4, we see the top satellite view of the bike storage environment. The red area indicates where the bike is not allowed to park, the green area represents where the bike is allowed to park, and the black area is where the bike can not be parked.

The blue dots in fig. 9.4 indicate the measurement locations. The same location was tested multiple times by rotating the bike by 90 degrees, having 4 measurements per location. We deployed 8 Anchor nodes (*D*-*K*) in the bike storage. This number was chosen as it complies with the node density required by Skopei for the autonomous bike-sharing solution.

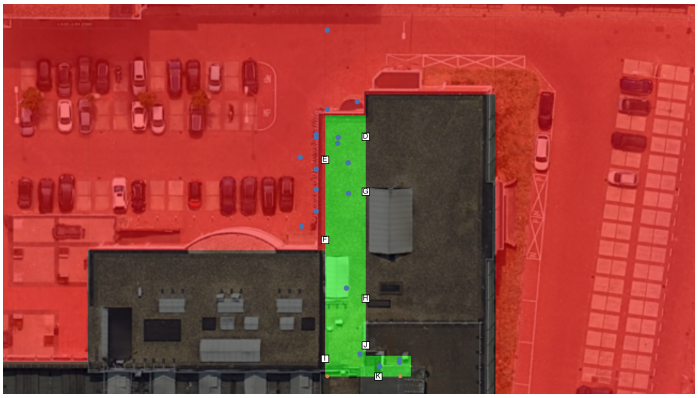


Figure 9.4: **Satellite overview of the bike storage**

In total, 59 locations were tested, and each node attempted to make 6 repetitions of measurement in both the reflector and initiator roles. The total number of distance measurements then becomes:

$$\begin{aligned} N_{measurements} &= N_{repetitions} \cdot N_{location} \cdot N_{roles} \cdot N_{anchors} \\ 5664 &= 6 \cdot 59 \cdot 2 \cdot 2 \end{aligned} \tag{9.2}$$

Due to the harsh no line of sight conditions, out of these 5664, 3890 measurements failed.

9.4 Raw measurements data

This section presents the raw measurement data before any preprocessing is done. We analyze the data to get an idea of why certain patterns are present and what implication these patterns have on the subsequent distance and confidence estimators. We analyze RSSI and the Nordic-specific distance estimations from eq. (5.5) [34].

9.4.1 RSSI

In fig. 9.5, we see the RSSI values on the y-axis and the reference distance on the x-axis. The blue line in the middle represents the theoretical path loss, as per eq. (2.1).

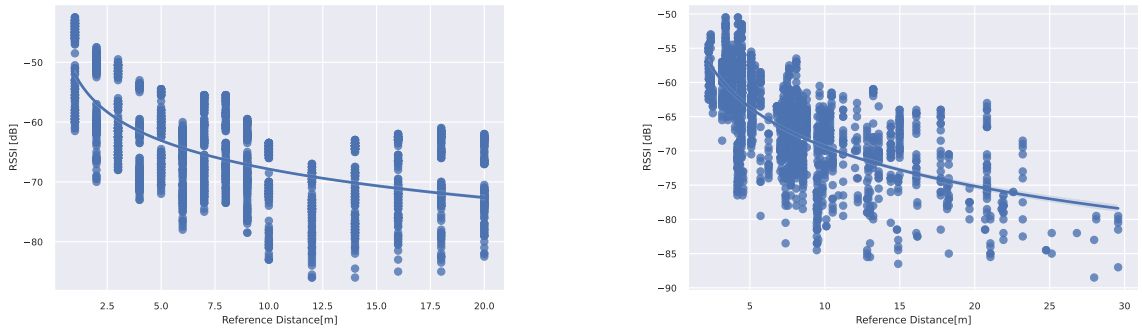
The graph on the left is the open-field experiment, and the graph on the right is the bike storage.

In fig. 9.5, we show that the RSSI is very volatile when the distance is the same. Note that the spread is 20dB for both the open field and the bike storage. As we rotate the bike lock at each step in both scenarios, we get this pattern due to multipath effects. Multipath fading changes the RSSI significantly.

The bike storage graph also has many more points close to zero meters, while the points in the open field are nicely distributed over the x-axis. This is because measurements close to zero meters are less likely to fail, and we only have a few measurements where the distance between the anchor nodes is large.

9.4.2 Phase slope

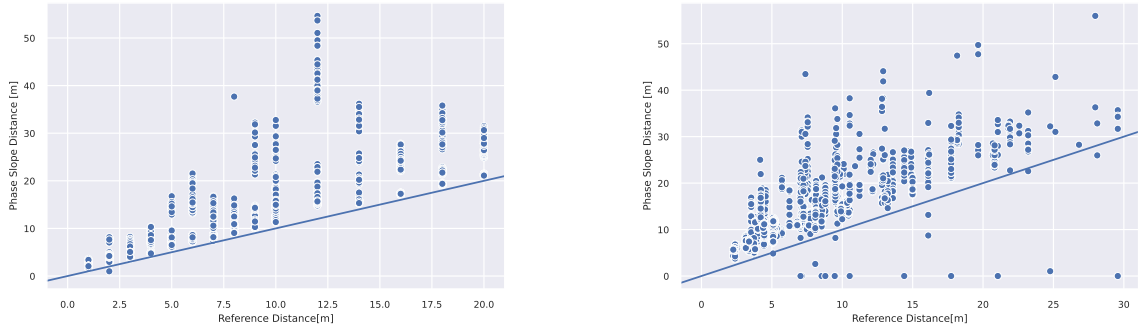
In fig. 9.6, we see the phase slope versus the reference distance. We first notice that the reference distance (the straight blue line) is always more than the phase slope distance for the open field



Open field **Bike storage**

Figure 9.5: **RSSI vs reference distance**

measurements.



Open field **Bike storage**

Figure 9.6: **Phase slope vs reference distance**

Most points are above the reference distance for the bike storage environment, but some are below the reference distance.

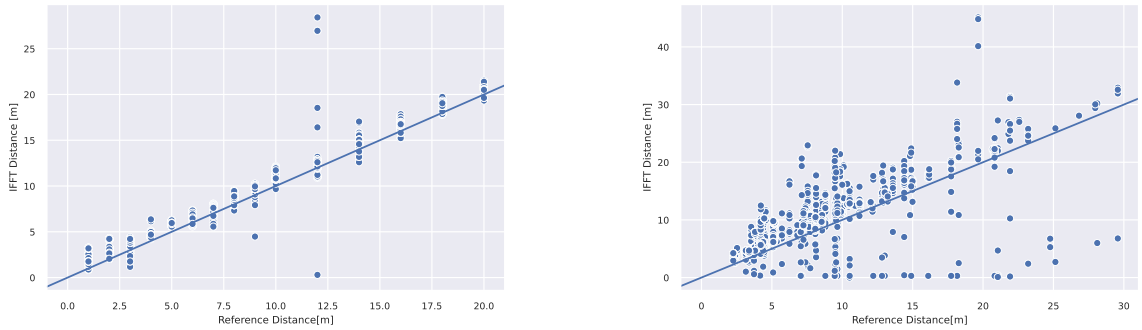
In the case of the open-field experiment, the estimations above the reference distance can be explained by the fact that, without interference from an external source, we either measure the direct path or the direct path plus reflections with the ground. Adding extra paths causes the phase slope to be more steep than it is for only the direct path, causing an overestimation of distance.

In the case of the bike storage experiment, we can have a distorted phaseshift due to interference by other $2.4GHz$, also making an underestimation possible.

9.4.3 IFFT distance

In fig. 9.7, we have a scatterplot for the ifft distance. The spread around the reference distance is minimal for the open field case compared to the RSSI and phase slope scatter plots.

The spread is quite constant, independent of whether the distance is 5 m or 20 m. In the IFFT distance estimator, we can separate the paths, as explained in section 6.4. Some points deviate only at 12 m. In this case, the conditions are such that the signal deconstructively interferes with itself and causes a higher spread in the measurements.



Open field **Bike storage**

Figure 9.7: **Phase slope vs reference distance**

For the distance estimators, the IFFT method outperforms the other methods, like the RSSI and phase slope distance estimators.

9.4.4 SINR

In fig. 9.8, we see the signal-to-noise plus interference ratio per anchor node for the distance 0 m to 20 m. For anchor node Juliett, the SINR is constant, independent of the distance, except for a small spike in at 12 m. At a distance of 12 m, we see that the SINR increases. This matches the less accurate measurements seen in section 6.4 at 12 m. For anchor node Kilo, the SINR gradually increases towards 12 m and then decreases again after the 12 m mark has been passed.

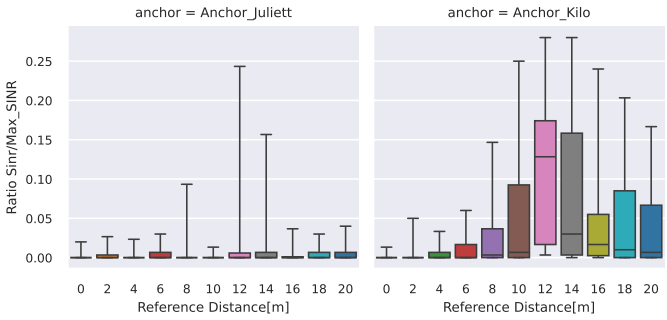


Figure 9.8: **Open field SINR vs reference distance**

The SINR for Kilo is much worse than Juliett because Kilo does not have a direct line of sight. For anchor Kilo, the circuit board blocks the line of sight with the bike lock, explaining the lower SINR.

In fig. 9.9, we see the SINR for all anchors D to K in the bike storage environment. Not every anchor has points on the entire x-axis because of the limited measurement points taken, shown in fig. 9.4 and limited successful meaurements. The noise level rises for most nodes when the distance increases, indicating a weaker connection.

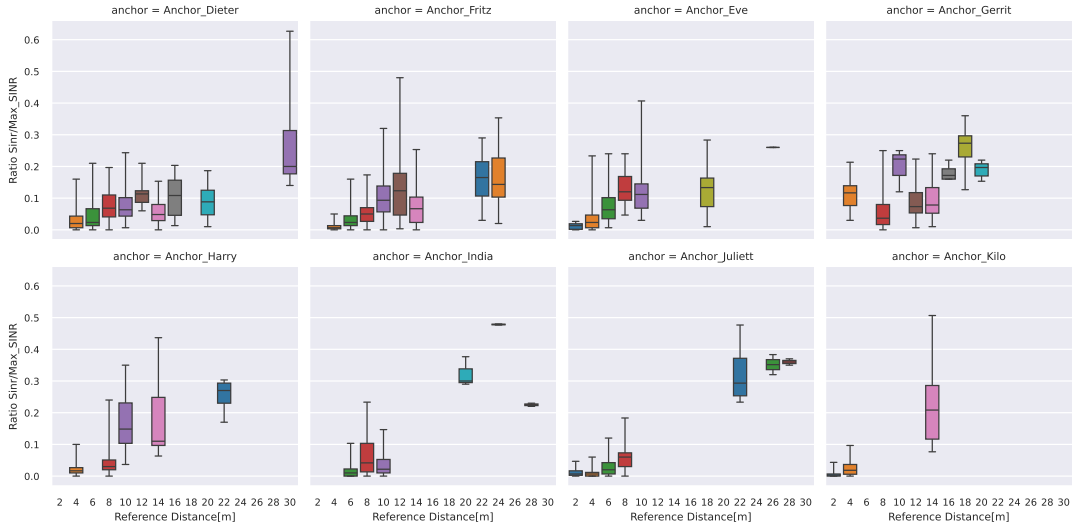


Figure 9.9: **Bike storage SINR vs reference distance**

9.4.5 Lock angle

We evaluate the lock angle to find the relation between distance estimation performance and the lock angle. In fig. 9.10, we see a histogram plot of the $ifft_{dist}$ error for the open field scenario. The different columns are the lock at different angles.

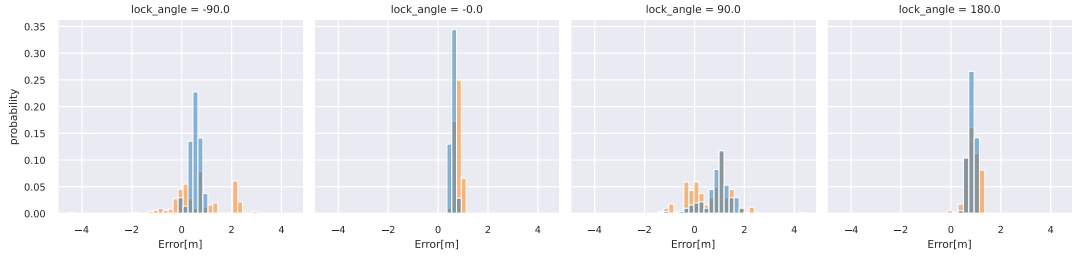


Figure 9.10: **Histplot of the lock angle**

We measure 4 distinct lock angles. The lock angle at 0 deg has the least spread, then the lock at 180 deg, and the lock at 90 deg and -90 deg have the most spread. This exemplifies the importance of lock angle to precision. With enough measurements, the ifft distance estimate can be accurate; if the number of measurements is limited, we must consider that the spread in certain conditions is significant.

9.5 Range estimation

The range estimator aims to estimate the distance and confidence based on the preprocessed measurement data. Ideally, we want our distance estimation to be accurate and precise, removing the need for a confidence level. However, as shown in section 9.4, in the challenging conditions of the

bike storage environment, we need to identify the poor and good measurements to get optimal performance.

We consider the distance estimation separately from the confidence estimation.

9.5.1 Distance estimation

The distance estimator tries to find the best fitting distance considering the set of parameters from preprocessed measurement vector in eq. (5.5).

Feature selection

We tested several model implementations to find which feature set performs best using the method described in section 6.5.3.

We compare six implementations using the cross-validation scheme explained in section 6.5.6. Every measurement belongs to a measurement set $\mathbf{D}$ of eq. (4.2). Repetitions of the same measurement in matrix $\mathbf{D}$ can be very similar. To prevent overfitting of the machine learning model, we do not mix measurements belonging to the same measurement matrix $\mathbf{D}$ in the test and train set.

The implementations use the $ifft_{dist}$ without and with bias compensation and Multi-layer Perceptron machine learning models with different input features. In fig. 9.11, we see the empirical cumulative distribution function of the mean absolute error of 6 different distance estimator implementations.

The implementations are as follows:

- **Method 0** IFFT estimation without bias compensation
- **Method 1** IFFT estimation with bias compensation
- **Method 2** MLP model estimation with features $[ifft_{dist}]$
- **Method 3** MLP model estimation with features $[ifft_{dist}, Rssi_{avg}]$
- **Method 4** MLP model estimation with features $[ifft_{dist}, Rssi_{avg}, Anchor_1, \ldots, Anchor_k]$
- **Method 5** MLP model estimation with features $[ifft_{dist}, Rssi_{avg}, Anchor_1, \ldots, Anchor_k, Phase\_slope_{dist}]$

The features $Anchor_1$ to $Anchor_k$ are binary encoded as either a 1 or 0, depending on if the node was involved in the measurement.

In fig. 9.11, we see that the performance of method 0 is the worst. This is to be expected, as it does not have any way to compensate for the bias present in the $ifft_{dist}$ measurements.

Method 2, 3, and 4 perform approximately equal up to an error of 3 m. After that, the machine learning implementations outperform the bias-compensated $ifft_{dist}$. This happens because the $ifft_{dist}$ does not consider multipath reflections for distance estimation. The machine learning method can compensate better when the error is higher because it is learned from example data.

Methods 5 and 6 always outperform the other methods. Method 5 has the added information that knows which anchor node the measurement is coming from. With this extra information, it can better predict distance using the $Rssi$ and $ifft_{dist}$ specific to that anchor node. Method 6 also has the $phase\_slope_{dist}$ as an input feature. The phase slope can better distinguish the line of sight from no line of sight conditions, again giving extra information about whether the $ifft_{dist}$ needs compensation.

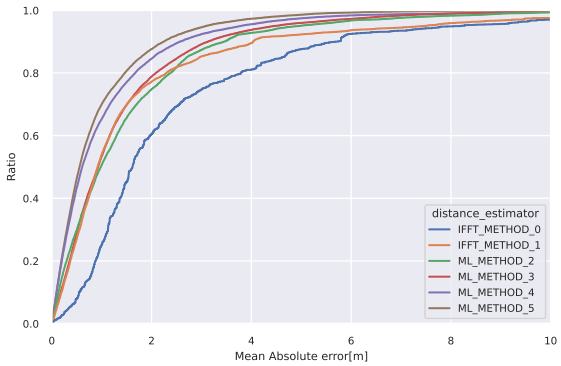
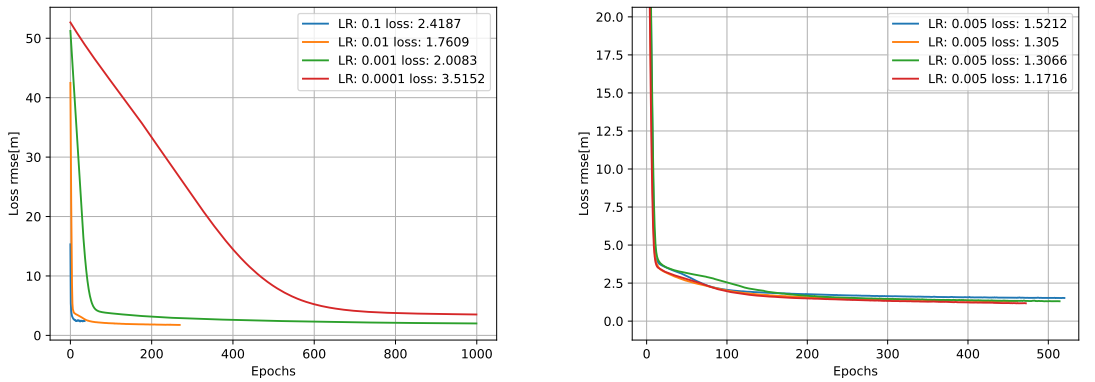


Figure 9.11: **ECDF distance estimation method comparison**

Hyper parameters

Besides selecting a set of features, our MLP model has several hyperparameters that must be adjusted for optimal performance. We will now evaluate the effect of the model performance based on the settings of these hyperparameters.

In fig. 9.12, we see a graph of the loss on the y-axis vs. epochs on the x-axis using the *adam* optimization algorithm [23]. In the graph on the left, we vary the learning rate. We see that the learning rate greatly impacts the progression and final value of the loss. A low learning rate causes the model to take many epochs to reach a minimum, while a high learning rate causes the model to oscillate quickly. The optimum is somewhere in between. We found empirically that a learning rate of 0.005 generally leads to the best results.



Different learning rate **Constant learning rate differently initialized**

Figure 9.12: **The effect of the learning rate and initialization**

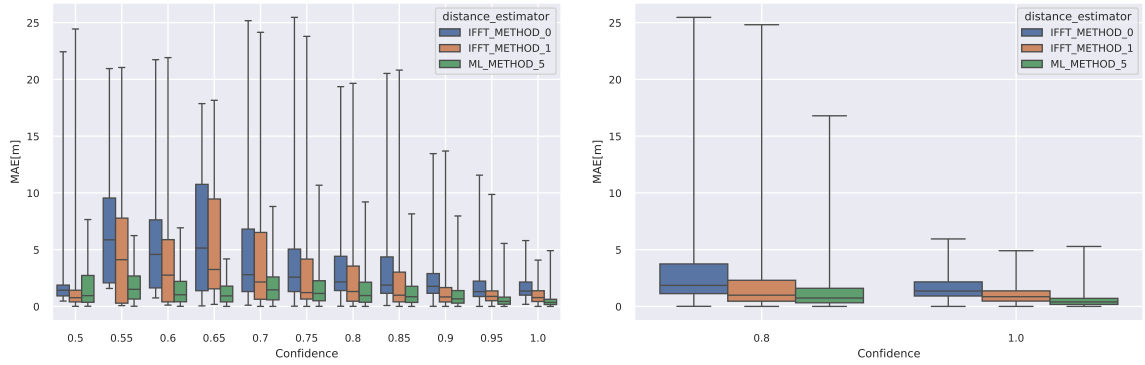
In the graph on the right in fig. 9.12, we compare a constant learning rate but with different model initialization. When we optimize our model, the weights and training batches are set randomly. The loss curve is much more consistent compared to the different learning rates. However, the final loss still differs by ≈ 0.35. To overcome this variability in performance, we test our model many times as

described in section 6.5.6 and look at whether it performs better on average. Also, as MLP supports online training, the model can retrain over time if we gather more measurement data.

9.5.2 Confidence estimation

We provided 3 confidence level estimators in section 6.6: *Rssi thresholding*, *SINR* and *Quality indicator* confidence estimation. The RSSI thresholding functions as a measurement filter. It assigns confidence 0 to all measurements when the threshold is reached.

The QI and SINR confidence estimators aim to provide low confidence when the error is likely high and high confidence when the error is likely low. We visualize this in the boxplot shown in fig. 9.13.



SINR **Quality indicator**

Figure 9.13: **Confidence estimation**

If we go from a high to low confidence, the distance between the first and third quartile increases until a confidence level of 0.75. After that, the confidence level stabilizes for the ML method, while the $ifft_{dist}$ methods keep increasing. At the lowest confidence levels, the MAE decreases again for all methods. Due to the small sample size, we have a reduced chance of an over or underestimation, explaining the reduced error.

The quality indicator is the binary counterpart of the SINR approach. The high confidence has a much lower spread in MAE than the lower confidence, making it a useful indicator of confidence.

9.6 Position estimation

The position estimator aggregates the distance and confidence estimations from the range estimator. Then, the position estimator finds the position and classifies if it has been parked correctly as described in chapter 7. We evaluate the performance of the position estimator for the classification and position estimation separately.

9.6.1 Position error

We define the estimated position error as the Euclidean distance between the estimated and actual positions. In table 9.1, we see an overview of the position estimation error for different algorithm configurations when the bike is parked inside. We compare the machine learning approach with the $ifft_{bias}$ and $ifft_{comp}$. The $ifft_{bias}$ uses the raw $ifft_{dist}$, while the $ifft_{comp}$ uses the train set to

find a bias, which is then used on the test set for verification. We can fairly compare the machine learning and $ifft_{dist}$ approaches by calculating the bias only based on the train set. Because total measurement set is limited, randomly drop 50% of our measurements to create more randomness between iterations.

Table 9.1: **MAE[m] for different range estimator configurations**

		Confidence Estimator			
		NONE	RT	QI	SINR
Distance Est.	$IFFT_{biased}$	3.80	3.52	2.98	2.66
	$IFFT_{comp}$	2.19	1.79	1.95	1.76
	ML	1.80	1.79	1.73	1.68

After optimizing for the best performance, we set the algorithm up in the following configuration. The $SINR_{max}$ was set to 0.1, meaning that measurements with more than 10% of the maximum possible SINR have a confidence of 0 assigned. The Quality indicator confidence estimator gives 0.2 confidence for *Bad* measurements and 1 for *good* measurements. The following feature set was used: [$ifft_{dist}$, $Rssi_{avg}$, $Anchor_1$, ... , $Anchor_k$, $Phase\_slope_{dist}$]. The ML method uses an MLP regressor. The learning rate was set to 0.005, we use a single hidden layer of 20 neurons with the *Relu* activation function and the *adam* optimizer. The rest of the hyperparameters were kept at their standard setting (see [49]).

We see that the machine-learning approach always outperforms both $ifft$ approaches. The confidence estimation gives a small improvement in the position estimation error for the ML approach. Our ML distance estimation is robust even if $ifft_{dist}$ predictions are poor. It can achieve this by combining the data of the Rssi, phase slope, and involved anchor. The $ifft_{dist}$ estimators lack this context and hence need the confidence estimation to compensate for poor measurements.

In fig. 9.14, we see that the first and third quartiles are always lower for the machine learning approach. This boxplot also shows the performance when the bike is parked inside. The performance outside is not of interest as this parking area is unallowed and has limited coverage by the network of routers.

9.6.2 Classification error

In table 9.2, we see the classification performance for different configurations of the range estimator. We have two values for every configuration of the range estimator: *Allowed* and *!Allowed*. *Allowed* represents the percentage of times the bike is correctly predicted to be inside, and *!Allowed* is the percentage of times the bike is correctly predicted to be outside.

The ML distance estimator performs best for correctly predicting that the bike is parked inside. The higher this value is, the fewer *obidient* users are incorrectly informed they parked in a not allowed area. *Disobidient* user are, however, more likely to get away with their wrongly parked bike, as in 13.7% the ML predicts the bike to be parked inside while it is outside.

Fig. 9.15 visualizes the error.

On the right, we see that the position estimation is just outside of the bike storage when the reference position is inside. The user could move their bike a little bit and trigger a new localization request, and now there is a good chance it will be classified as inside the bike storage.

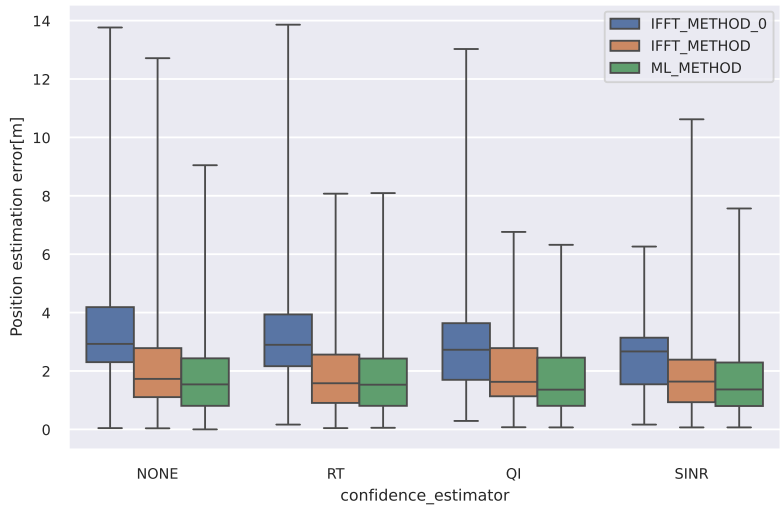


Figure 9.14: **Estimated position error [m] for different algorithm configurations**

Table 9.2: **Classification of park state, predicted correctly in %**

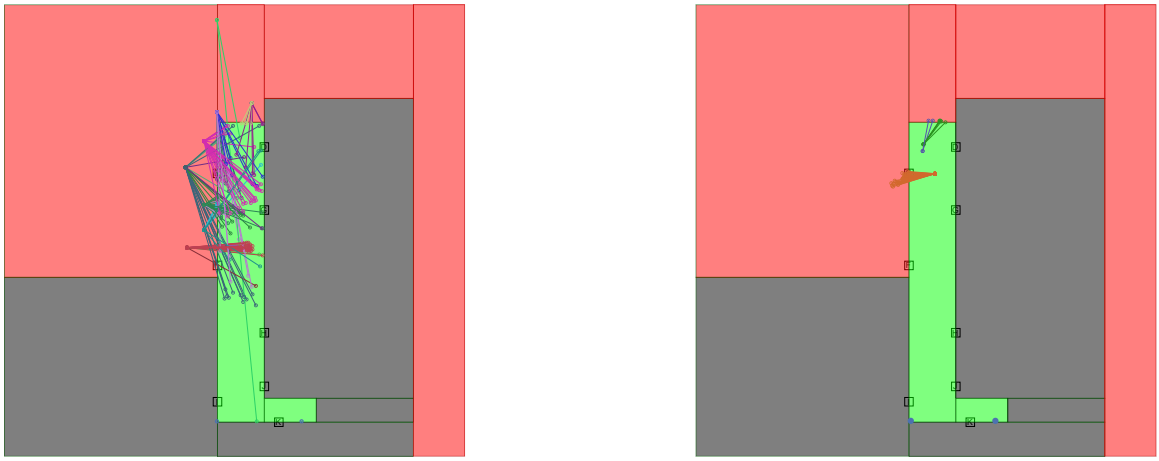
			Confidence Est.			
		Ref park state	NONE	RT	QI	SINR
Distance Est.	$IFFT_{biased}$	Allowed	75.99	74.12	80.2	81.61
		!Allowed	93.29	94.09	93.11	91.94
	$IFFT_{comp}$	Allowed	95.31	95.45	96.48	95.89
		!Allowed	86.03	89.17	84.78	87.82
	ML	Allowed	96.19	95.45	97.8	98.09
		!Allowed	88.99	87.2	88.36	86.3

For outside, we have many more cases where the bike is predicted to be parked inside. We often see that the bike is predicted to be right next to an anchor. Despite being so close, this anchor probably had no successful distance estimations. The position estimation algorithm does not consider that it is probably far away when an anchor has no successful estimation. This causes these unlikely estimations. This problem can be fixed by including anchors with no successful estimation in the position estimation, but that is outside this work's scope.

9.7 Limitations

We have shown that it is possible to find the location of the bike node inside a bike storage environment using the hardware already deployed for existing bike sharing of Skopei.

More tests are required to test if the solution presented in this thesis complies with the other requirements set in chapter 8. The implications of, for example, the power usage of distance measurements is still an open question that we can not answer with the current measurement data.



Reference outside **Reference inside**

Figure 9.15: **Incorrectly classified points by the ML method**

Chapter 10

Conclusions

The thesis successfully demonstrated the feasibility and effectiveness of the proposed RF-based indoor localization system, particularly in the context of bike-sharing services. The findings included detailed analyses of various measurement methods, such as RSSI, phase slope, and IFFT, and their performance in scenarios like open fields and bike storage environments.

The research made significant technical contributions, including developing algorithms for distance and position estimation and the innovative use of the Nordic Distance Measurement library for precise measurements. It also explored the impact of environmental factors and device orientations on measurement accuracy.

We employed a machine learning algorithm to find patterns in the measurement data due to complex multipath fading effects.

The study acknowledged the limitations and challenges encountered, such as synchronization issues, the impact of environmental factors on signal quality, and the limitations of the hardware used.

The practical applications of the research in the context of bike-sharing services were highlighted, showing potential for improving operational efficiency and user experience.

In the bike storage test case, our proposed solution achieved a mean absolute error in position estimation of $1.68m$ compared to $3.80m$ of a naive approach. The classification accuracy improved from 75.99% for the naive approach to 98.09% with our proposed machine-learning-based approach.

10.1 Future work

Within the limited timespan of this thesis, we aimed to gain as much valuable knowledge about the problem as possible. This means that some assumptions must be made for the work to remain within a specific scope. For future work, we advise the following research directions:

- **Included failed measurements** A more complex error function that includes failed measurements can improve position estimation performance
- **Automate measurements** To keep the system deployment scalable, lidar would allow us to gather more training data automatically. This extra training data can be used for an improved machine learning model.
- **IMU integration** Integrating IMU data in the measurement reports allows us to find the bike's orientation. We have shown that the orientation has a great impact on the measurement.

Hence, using this orientation as input for the machine learning algorithm could greatly improve performance.

- **More measurement** To check for robustness, we need to find how well the solution performs under different conditions. In this research, we have laid the foundation for data gathering, which can be used in future work to gather more samples in different environments.
- **Advanced Environment description** A more advanced environment description used by the distance and position estimator could greatly improve performance, as it becomes possible to compensate for brick walls that the signal has to travel around.
- **Model tuning** In this research, we have already shown that model (hyper)parameters greatly influence the performance of distance estimation. Better tuning with more data and an advanced environment description could yield even better results on distance estimation.

Bibliography

[1] Ali Abedi and Deepak Vasisht. Non-cooperative wi-fi localization & its privacy implications. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 570–582, Sydney NSW Australia, October 2022. ACM.

[2] C. Alippi and G. Vanini. A RSSI-based and calibrated centralized localization technique for-Wireless Sensor Networks. In *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)*, pages 301–305, Pisa, Italy, 2006. IEEE.

[3] Aline Baggio and Koen Langendoen. Monte Carlo localization for mobile wireless sensor networks. *Ad Hoc Networks*, 6(5):718–733, July 2008.

[4] Primož Bencak, Darko Hercog, and Tone Lerher. Indoor Positioning System Based on Bluetooth Low Energy Technology and a Nature-Inspired Optimization Algorithm. *Electronics*, 11(3):308, January 2022.

[5] Alan Bensky. Technologies and applications. In *Short-range Wireless Communication*, pages 387–430. Elsevier, 2019.

[6] Pepijn Boer, Jac Romme, Jochem Govers, and Guido Dolmans. Performance of High-Accuracy Phase-Based Ranging in Multipath Environments. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–5, Antwerp, Belgium, May 2020. IEEE.

[7] Giuseppe Bonaccorso. *Machine learning algorithms: a reference guide to popular algorithms for data science and machine learning*. Packt, Birmingham Mumbai, 2017.

[8] Fuhu Che, Qasim Zeeshan Ahmed, Jaron Fontaine, Ben Van Herbruggen, Adnan Shahid, Eli De Poorter, and Pavlos I. Lazaridis. Feature-Based Generalized Gaussian Distribution Method for NLoS Detection in Ultra-Wideband (UWB) Indoor Positioning System. *IEEE Sensors Journal*, 22(19):18726–18739, October 2022.

[9] Hsieh-Chung Chen, Tsung-Han Lin, H. T. Kung, Chit-Kwan Lin, and Youngjune Gwon. Determining RF angle of arrival using COTS antenna arrays: A field evaluation. In *MILCOM 2012 - 2012 IEEE Military Communications Conference*, pages 1–6, Orlando, FL, USA, October 2012. IEEE.

[10] Tuochao Chen, Justin Chan, and Shyamnath Gollakota. Underwater 3D positioning on smart devices. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 33–48, New York NY USA, September 2023. ACM.

[11] Chenshu Wu, Zheng Yang, Yunhao Liu, and Wei Xi. WILL: Wireless Indoor Localization without Site Survey. *IEEE Transactions on Parallel and Distributed Systems*, 24(4):839–848, April 2013.

[12] Silvano Cortesi, Christian Vogt, and Michele Magno. Comparison between an RSSI- and an MCPD-Based BLE Indoor Localization System. *Computers*, 12(3):59, March 2023.

[13] Paul DeMaio. Bike-sharing: History, Impacts, Models of Provision, and Future. *Journal of Public Transportation*, 12(4):41–56, December 2009.

[14] Bosch DIY. PLR 30 C Digital Laser Measure | Bosch DIY. https://www.bosch-diy.com/gb/en/p/plr-30-c-0603672100.

[15] Mauricio A. Caceres Duran, Antonio A. D'Amico, Davide Dardari, Mats Rydström, Francesco Sottile, Erik G. Ström, and Lorenzo Taponecco. Terrestrial Network-Based Positioning and Navigation. In *Satellite and Terrestrial Radio Positioning Techniques*, pages 75–153. Elsevier, 2012.

[16] Georg Fischer, Joan Bordoy, Dominik Jan Schott, Wenxin Xiong, Andrea Gabbrielli, Fabian Hoflinger, Kai Fischer, Christian Schindelhauer, and Stefan Johann Rupitsch. Multimodal Indoor Localization: Fusion Possibilities of Ultrasonic and Bluetooth Low-Energy Data. *IEEE Sensors Journal*, 22(6):5857–5868, March 2022.

[17] S. Gezici and H.V. Poor. Position Estimation via Ultra-Wide-Band Signals. *Proceedings of the IEEE*, 97(2):386–403, February 2009.

[18] Mohammad Reza Gholami, Henk Wymeersch, Erik G Ström, and Mats Rydström. Wireless network positioning as a convex feasibility problem. *EURASIP Journal on Wireless Communications and Networking*, 2011(1):161, December 2011.

[19] Yan Huang, Jianying Zheng, Yang Xiao, and Miao Peng. Robust Localization Algorithm Based on the RSSI Ranging Scope. *International Journal of Distributed Sensor Networks*, 11(2):587318, February 2015.

[20] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer Texts in Statistics. Springer US, New York, NY, 2021.

[21] Hyeon Jo and Seungku Kim. Indoor Smartphone Localization Based on LOS and NLOS Identification. *Sensors*, 18(11):3987, November 2018.

[22] Hyung-Sin Kim, Sam Kumar, and David E. Culler. Thread/OpenThread: A Compromise in Low-Power Wireless Multihop Network Architecture for the Internet of Things. *IEEE Communications Magazine*, 57(7):55–61, July 2019.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980 [cs].

[24] Manon Kok, Frida Viset, and Mostafa Osman. A Framework for Indoor Localization Using the Magnetic Field. In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, pages 385–387, Paphos, Cyprus, June 2022. IEEE.

[25] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. SpotFi: Decimeter Level Localization Using WiFi. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 269–282, London United Kingdom, August 2015. ACM.

[26] Praveen Kumar, Lohith Reddy, and Shirshu Varma. Distance measurement and error estimation scheme for RSSI based localization in Wireless Sensor Networks. In *2009 Fifth International Conference on Wireless Communication and Sensor Networks (WCSN)*, pages 1–4, Allahabad, India, December 2009. IEEE.

[27] Chenglong Li, Emmeric Tanghe, David Plets, Pieter Suanet, Jeroen Hoebeke, Eli De Poorter, and Wout Joseph. ReLoc: Hybrid RSSI-and Phase-based Relative UHF-RFID Tag Localization with COTS Devices. *IEEE Transactions on Instrumentation and Measurement*, pages 1–1, 2020.

[28] Wenjie Luo, Qun Song, Zhenyu Yan, Rui Tan, and Guosheng Lin. Indoor Smartphone SLAM with Learned Echoic Location Features, October 2022. arXiv:2210.08493 [cs, eess].

[29] Hongjiang Lyu, Linghe Kong, Chengzhang Li, Yunxin Liu, Jiansong Zhang, and Guihai Chen. BikeLoc: a Real-time High-Precision Bicycle Localization System Using Synthetic Aperture Radar. In *Proceedings of the First Asia-Pacific Workshop on Networking*, pages 57–63, Hong Kong China, August 2017. ACM.

[30] Rainer Mautz. Overview of current indoor positioning systems. *Geodesy and Cartography*, 35(1):18–22, January 2009.

[31] Ahasanun Nessa, Bhagawat Adhikari, Fatima Hussain, and Xavier N. Fernando. A Survey of Machine Learning for Indoor Positioning. *IEEE Access*, 8:214945–214965, 2020.

[32] Jiazhi Ni, Fusang Zhang, Jie Xiong, Qiang Huang, Zhaoxin Chang, Junqi Ma, BinBin Xie, Pengsen Wang, Guangyu Bian, Xin Li, and Chang Liu. Experience: pushing indoor localization from laboratory to the wild. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 147–157, Sydney NSW Australia, October 2022. ACM.

[33] Nordic. Measuring distance with the Nordic Distance Toolbox.

[34] NordicSemi. Distance Measurement. https://docs.nordicsemi.com/bundle/ncs-latest/page/nrfxlib/nrf_dm/README.html#nrf-dm.

[35] NordicSemi. nRF52840 Dongle. https://www.nordicsemi.com/Products/Development-hardware/nRF52840-Dongle.

[36] Mathias Pelka, Christian Bollmeyer, and Horst Hellbruck. Accurate radio distance estimation by phase measurements with multiple frequencies. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 142–151, Busan, South Korea, October 2014. IEEE.

[37] Pradini Puspitaningayu, Nobuo Funabiki, Yuanzhi Huo, Kazushi Hamazaki, Minoru Kuribayashi, and Wen-Chung Kao. Application of Fingerprint-based Indoor Localization System Using IEEE 802.15.4 to Two-Floors Environment. In *2022 IEEE 4th Global Conference on Life Sciences and Technologies (LifeTech)*, pages 239–240, Osaka, Japan, March 2022. IEEE.

[38] Lanxin Qiu, Zhangqin Huang, Shaohua Zhang, Cheng Jing, Hao Li, and Shuyao Li. Multifrequency Phase Difference of Arrival Range Measurement: Principle, Implementation, and Evaluation. *International Journal of Distributed Sensor Networks*, 11(11):715307, November 2015.

[39] Anshul Rai, Krishna Kant Chintalapudi, Venkata N. Padmanabhan, and Rijurekha Sen. Zee: zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 293–304, Istanbul Turkey, August 2012. ACM.

[40] Mark A. Richards, James A. Scheer, and William A. Holm. *Principles of modern radar*. SciTech publ, Raleigh (N.C.), 2010.

[41] Hamada Rizk, Ahmed Elmogy, and Hirozumi Yamaguchi. A Robust and Accurate Indoor Localization Using Learning-Based Fusion of Wi-Fi RTT and RSSI. *Sensors*, 22(7):2700, March 2022.

[42] Samer S. Saab and Zahi S. Nakad. A Standalone RFID Indoor Positioning System Using Passive Tags. *IEEE Transactions on Industrial Electronics*, 58(5):1961–1970, May 2011.

[43] Cung Lian Sang, Bastian Steinhagen, Jonas Dominik Homburg, Michael Adams, Marc Hesse, and Ulrich Rückert. Identification of NLOS and Multi-Path Conditions in UWB Localization Using Machine Learning Methods. *Applied Sciences*, 10(11):3980, June 2020.

[44] Yannic Schroder, Dennis Reimers, and Lars Wolf. Accurate and Precise Distance Estimation from Phase-Based Ranging Data. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8, Nantes, September 2018. IEEE.

[45] Yannic Schröder and Lars Wolf. InPhase: Phase-based Ranging and Localization. *ACM Transactions on Sensor Networks*, 18(2):1–39, May 2022.

[46] Nordic Semi. Nordic Distance Measurement library. https://docs.nordicsemi.com/bundle/ncs-latest/page/nrfxlib/nrf_dm/doc/nrf_dm_overview.html.

[47] Souvik Sen, Jeongkeun Lee, Kyu-Han Kim, and Paul Congdon. Avoiding multipath to revive inbuilding WiFi localization. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 249–262, Taipei Taiwan, June 2013. ACM.

[48] Yuanchao Shu, Cheng Bo, Guobin Shen, Chunshui Zhao, Liqun Li, and Feng Zhao. Magicol: Indoor Localization Using Pervasive Magnetic Field and Opportunistic WiFi Sensing. *IEEE Journal on Selected Areas in Communications*, 33(7):1443–1457, July 2015.

[49] sklearn. sklearn.neural_network.MLPRegressor.

[50] Colin Stagner, Andrew Conrad, Christopher Osterwise, Daryl G. Beetner, and Steven Grant. A Practical Superheterodyne-Receiver Detector Using Stimulated Emissions. *IEEE Transactions on Instrumentation and Measurement*, 60(4):1461–1468, April 2011.

[51] Mihai-Ionut Stanciu, Jerome Brillant, Claudio Rey, and Khurram Waheed. Accurate Distance Measurement Using Narrowband Systems. In *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 937–940, Lansing, MI, USA, August 2021. IEEE.

[52] Johan Nicolas Suarez Lojan. *Improving Accuracy of MCPD Distance Measurements*. PhD thesis, NTNU, 2023.

[53] U-Blox. BMD-34 series (open CPU), July 2019. https://www.u-blox.com/en/product/bmd-34-series-open-cpu.

[54] Arthur Venon, Yohan Dupuis, Pascal Vasseur, and Pierre Merriaux. Millimeter Wave FMCW RADARs for Perception, Recognition and Localization in Automotive Applications: A Survey. *IEEE Transactions on Intelligent Vehicles*, 7(3):533–555, September 2022.

[55] Chia-Cheng Wang, Jyh-Cheng Chen, Yi Chen, Rui-Heng Tu, Jia-Jiun Lee, Yu-Xin Xiao, and Shan-Yu Cai. MVP: magnetic vehicular positioning system for GNSS-denied environments. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 531–544, New Orleans Louisiana, October 2021. ACM.

[56] Weiguo Wang, Luca Mottola, Yuan He, Jinming Li, Yimiao Sun, Shuai Li, Hua Jing, and Yulei Wang. MicNest: Long-Range Instant Acoustic Localization of Drones in Precise Landing. In *Proceedings of the Twentieth ACM Conference on Embedded Networked Sensor Systems*, pages 504–517, Boston Massachusetts, November 2022. ACM.

[57] Yan Wu, Niko Joram, and Rainer T Hach. Quantitative Comparison of Distance Estimation Performance between TLS-MP and iFFT Methods in IEEE 802.15.4a Channel Models Using PSFM Radar Technique. In *2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7, Lloret de Mar, Spain, November 2021. IEEE.

[58] Xu Yang, Yuqing Yin, Zhaoyu Sun, Shouwan Gao, and Qiang Niu. Autonomous Passive Localization Algorithm for Shared Bikes. *IEEE Access*, 7:119917–119930, 2019.

[59] Moustafa Youssef and Ashok Agrawala. The Horus WLAN location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218, Seattle Washington, June 2005. ACM.

[60] Tong Yu and Hong Zhu. Hyper-Parameter Optimization: A Review of Algorithms and Applications, March 2020. arXiv:2003.05689 [cs, stat].

[61] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. A Survey of Indoor Localization Systems and Technologies. *IEEE Communications Surveys & Tutorials*, 21(3):2568–2599, 2019.

[62] Pouria Zand, Jac Romme, Jochem Govers, Frank Pasveer, and Guido Dolmans. A high-accuracy phase-based ranging solution with Bluetooth Low Energy (BLE). In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8, Marrakesh, Morocco, April 2019. IEEE.

[63] Farzaneh Zangenehnejad and Yang Gao. GNSS smartphones positioning: advances, challenges, opportunities, and future perspectives. *Satellite Navigation*, 2(1):24, November 2021.

[64] Tingwei Zhang, Peng Zhang, Paris Kalathas, Guangxin Wang, and Huaping Liu. A Machine Learning Approach to Improve Ranging Accuracy with AoA and RSSI. *Sensors*, 22(17):6404, August 2022.

[65] Jin Zheng, Kailong Li, and Xing Zhang. Wi-Fi Fingerprint-Based Indoor Localization Method via Standard Particle Swarm Optimization. *Sensors*, 22(13):5051, July 2022.