Delft University of Technology
Master of Science Thesis in Embedded Systems

Sonic Filter Localization

Integrating Particle Filter with Sound Source Localization techniques to achieve accurate indoor localization

Wesley de Hek

Networked Systems

TU Delft Delft University of Technology

Sonic Filter Localization

Integrating Particle Filter with Sound Source Localization techniques to achieve accurate indoor localization

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

Wesley de Hek
4975928
11-06-2024

Author
Wesley de Hek
4975928

Title
Sonic Filter Localization: Integrating Particle Filter with Sound Source Localization techniques to achieve accurate indoor localization

MSc Presentation Date
26-06-2024

Graduation Committee

Prof. Dr. R. V. Prasad	Delft University of Technology
Dr. C. Lofi	Delft University of Technology
Dr. S. Narayana	Delft University of Technology

“Quiet people have the loudest minds” – Stephen Hawking

Preface

Most of my life I've spend in the pursuit of technology, by researching new technologies and applying them in practice. Here my fundamental interest lay with software development and more specifically developing software for embedded devices. Because of this, an education in the direction of Embedded Systems made the most sense to me, which resulted in a bachelor's degree in Software Engineering followed by a master in Embedded Systems. This thesis represents the culmination of years of dedicated study, research, and countless hours of hard work during the past few years of my study at the TU Delft.

This work wouldn't have been possible if it wasn't for the people around me. Because of this, I would like to first and for most thank my girlfriend Dorien Sterrenburg for her unwavering support during the past few years of my study, she has been and will continue to be my greatest source of motivation and strength in life. Next to this, I would like to express my heartfelt appreciation to my advisor, Dr. Sujay Narayana, for his guidance and expertise, and to Professor Ranga Rao Venkatesha Prasadfor allowing me to do this research. Lastly, I would like to thank my parents for their support during all this.

In writing this thesis, I hope to contribute to the field of indoor localization, mainly focused on localizing robotic vehicles. I hope that my thesis will spark further discussions and encourage future research into the field of indoor localization. Hopefully this work will one day serve as a steppingstone for other students/researchers on a similar pursuit.

Wesley de Hek
Delft, June 2024

Contents

Abstract

For my master's thesis, I developed a novel indoor localization approach that can achieve accurate localization results. Indoor localization is a well-known topic of research, and many attempts have been made to find the so-called holy grail. For a robotic car swarm to be able to execute a specific task in a specific room, each robot in the swarm needs to know its own location with respect to a map of the building. Allowing the robot to deduce this location himself allows for total automation of the swarm and helps mitigate errors along the way due to faulty sensor readings. Numerous works have tried to solve this problem by using beacons or heavily trained machine learning networks. These methods, however, prevent the robot from working everywhere, as either the location needs to be adapted or the robot needs to be trained for the location. To mitigate these issues, the work of this thesis is focused on achieving indoor localization that works everywhere by researching if a combination of a particle filter and sound source localization techniques can achieve high-accuracy indoor localization.

To achieve this, particle filter and source localization techniques are combined based on probability theory. Here, so-called localization tables are used to estimate the robot's position based on the geometrical properties of the map, the DOA, and the distance from a received message. This table is then used to update the position and weight of the particles, which can then be used to make an educated guess of the robot's position. This approach proved to be effective, as it was able to achieve average RMSE results as low as **3.83 cm**, where the robot only needed to drive **363 cm** and localization was achieved within **9.52 seconds**. The research showed that deploying more cars in the swarm leads to better results, as fewer transmissions are needed, and less distance needs to be traveled. In addition, the research also showed that localization can be achieved while driving even less distance by sharing the localization tables between the robots. This, however, does come at the cost of localization accuracy, resulting in an RMSE of **19.89 cm** while driving only **158 cm**. To conclude this research, based on the results of the novel fusion of a particle filter and sound source localization techniques, it can be concluded that the work in this thesis offers a great contribution to the field of indoor localization using audio-based signals.

1

Introduction

A swarm of robots is a group of robots existing out of many robots that work together to achieve a larger common task, which they most likely wouldn't have been able to achieve individually. The main idea behind robotic swarms originates from nature, where social insects, like ants, wasps, bees, termites, etc. [1, 2] pool their workforce to achieve tasks that are impossible for one individual. Their combined power results in emergent behavior, which means that their collective being results in properties that one individual does not have on its own [3]. An example of this behavior can be seen with the Nasutitermes triodiae, a termite species that lives in Australia, who work together in groups of thousands to build massive so-called 'termite cathedrals' out of simple building materials like feces, saliva, and mud [4–6]. These cathedrals regulate the environmental conditions for their tunnels underneath the ground, allowing the terminates to live shielded from the outside world.

Just like in nature, robotic swarms use local communication to divide the work and achieve their common task [1], in contrast to centralized control often applied in centralized distributed robotic systems [7]. Swarm-based operations, both in nature and robotics, have three advantages. First up, **robustness**, meaning that the swarm will still be able to achieve the goal even when one or more nodes in the swarm are omitted because of failure. Secondly, **scalability**, meaning that the swarm should be able to work with a varying number of nodes. Lastly, **flexibility**, meaning that the swarm should be able to adapt to changes in the environment and still achieve its goal [8]. This also means that a characteristic of a robotic swarm should also be that it should work in any environment, even when that environment is prone to changes.

To achieve this, each robot should know which task it needs to execute and where it needs to do this, combined with either its own current location or a path to the task site. For swarms operating in outdoor environments, this can be achieved by using GPS for localization [9, 10], combined with relative localization when the robots need to work near each other [11]. However, this technique cannot be used for indoor scenarios due to the lack of GPS coverage inside a building. Most of the work on indoor localization is based on either radio frequency (RF) signals [12–19], light-based signals [20–27], or audio-based signals [28–31], where a lot of the work is dependent on specialized beacons (speakers, RF beacons) with known locations being present in the indoor environment. This means that the infrastructure of the building needs to be altered to achieve indoor localization, which voids the flexibility property of a robotic swarm, as these approaches do not allow the swarm to function in every environment. Next to this, a lot of the works, especially those based on audio-based localization, focus on localizing other objects in an indoor scenario rather than localizing itself [32–34]. To avoid these limitations and preserve the advantageous properties of robotic swarms, this thesis focuses on achieving indoor localization in any indoor environment. To achieve this, a combination of a particle filter (PF) and sound source localization (SSL) techniques is used, requiring only a building map to achieve indoor localization.

1.1. Motivation

As mentioned in the previous subsection, this thesis aims to achieve high accuracy indoor localization for robotic swarms using a combination of a particle filter and SSL techniques. Because of the unique nature of indoor environments, sound was chosen over RF and light-based as the preferred communication and localization technique. This choice was made because of the following reasons:

- **Wall attenuation:** sound-based signals, depending on the type of wall, hardly (or not at all) pass through it, due to the highly absorbing and reflecting nature of walls. This is in contrast to RF-based signals, which travel through walls with way less attenuation than audio-based signals. This phenomenon can be explained by the fact that RF signals have much smaller wavelengths (mm range instead of cm range), making it easier for these signals to travel through a wall [35]. The wavelength of a signal can be calculated using Equation 1.1, where λ is the wavelength in meters, c is the propagation speed of the wave in a certain medium given in m/s (approximately 343 m/s for sound signals and $3 * 10^8$ m/s for RF signals), and f is the frequency of the signal in Hz.

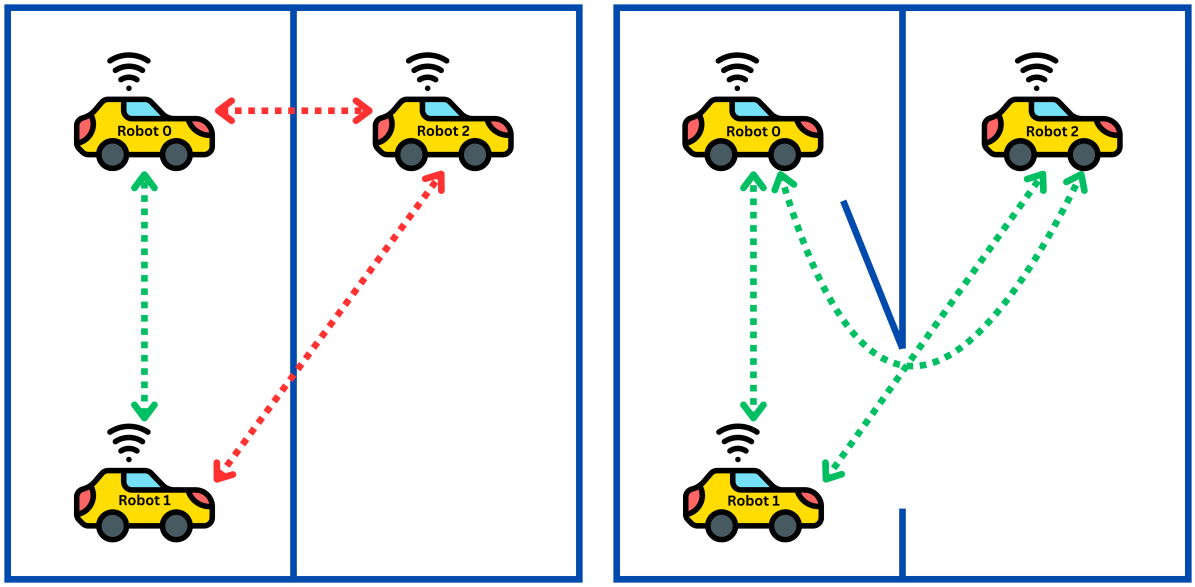
  $$\lambda = \frac{c}{f} \tag{1.1}$$

  Combined with the fact that low power sound signals will be used in this thesis, it can be assumed that the sound will not traverse walls in almost all situations. The fact that this doesn't happen is a desirable quality, as it prevents unnecessary communication between two robots who cannot contribute to their collective localization. This saves on processing resources by not processing inaccurate messages from robots on the other side of a wall. Such a scenario is sketched in Figure 1.1a, where two robots are separated by a wall and, therefore, cannot communicate with each other.

- **Bending around obstacles:** Sound has the unique property that it can bend around objects, unlike light that needs to reflect off an object to reach other places than direct line-of-sight. This sound property is called wave diffraction [36], which can approximately be modeled by Kirchhoff's diffraction formula. This property of sound gives two robots on opposite sides of a wall the opportunity to communicate with each other if there is an open door present in that wall. Such a scenario is sketched in Figure 1.1b, where robot 2 can communicate with robots 0 and 1 through the open door in the wall. As can be seen in Figure 1.1a, this communication becomes impossible when the door is closed since the sound is not able to traverse the wall. Because the sound travels through the door back up to the other robot, this also means that the direction from which the message is received will point in the direction of the door instead of in the direction where the other robot is. This is also a desirable quality, as it allows the robot to follow the sound through the door to the other robot without knowing anything about the environment. Another situation in which this property is advantageous is, in case of the situation on Figure 1.1b, when a message is received from below at a distance greater than the distance to the wall on the bottom. This tells us that the other robot must be in the room on the other side of that wall, which, when matched with a map, allows for self-localization if the situation is unique enough.

- **Lightning conditions:** Just like RF-based signals, sound-based signals can work under all lighting conditions, as they are not affected by the ambient light in the environment. This allows the robotic swarm to operate day and night, independent of the amount of light in the room. This is in contrast to communication using visible light, as this could be affected by numerous lightning conditions, like ambient light, shadows, etc.

- **Communication range:** The communication range of RF signals through the air is orders of magnitude higher than that of sound. This can be explained by the fact that air has a relatively low attenuation for RF frequencies compared to sound frequencies, resulting in a lower communication range. This is a desirable property for this project, as it prevents the communication between robots that are far apart and couldn't possibly produce accurate localization results. Therefore, being unable to communicate with these robots is an advantage as it saves on wasted processing power, allowing the robot to conserve energy.



(a) Robots not being able to communicate because of a wall.

(b) Robots able to communicate through the door.

Figure 1.1: Indoor communication scenarios.

A lot of work in the literature is focused on employing some form of artificial intelligence (AI) to achieve relative or indoor localization using sound-based signals [28, 37–45]. These approaches can achieve higher accuracy than most approaches that do not implement some form of AI. The use of AI, however, also comes with some disadvantageous:

- **Training data:** The biggest disadvantage of implementing AI is that it requires training data for every unique situation. Because of this, the robot cannot localize itself in a new environment without collecting training data first, voiding the flexibility constraint of robotic swarms. Next, any changes to the environment would also change the acoustic response, leading to diminishing results if no retraining is performed. This also means that for very dynamic environments, where, for example, a variable number of people are present, the results will fluctuate based on the number of people and their behavior.
- **Processing power:** Another disadvantage of employing AI is the amount of processing power needed to perform localization. This demand for processing power depends on the type of AI used and the amount of training data present. For this, it holds that the more training data, the more processing power is required, which is a realistic scenario if you want to localize in different locations. Not using an AI will, therefore, significantly reduce the processing power demand, allowing for more power-efficient robotic swarms.
- **Complexity:** Using AI is, compared to traditional approaches, infinitely more complex to implement and maintain. Refraining from using AI will thus make the code less complex and easier to understand for future maintainers.

Together with SSL, a particle filter is employed to achieve localization based on a building map. The choice for a particle filter was made because of its unique ability to represent and keep track of the probable position of the robot [46]. Next to this, the particle filter allows for error correction upon movement of the robot. This means that even in the worst-case scenario (when the SSL localizes the robot in a completely wrong position), the robot will still eventually know its true location after some movement through the building by eliminating all impossible particles on its path. All in all, the combination of SSL and particle filter offers a lightweight and robust way to localize robots in a robotic swarm in indoor environments.

1.2. Problem statement

In pursuit of the motivations stated above, it was clear that there is still room for improvement in the field of indoor localization using audio-based signals. At the time of writing this thesis and to the best of my knowledge, no work has ever been done before to combine a particle filter with SSL in a way as described here in this thesis. The following main problem statement, which will be answered in this thesis, arises from this:

How can a particle filter and sound source localization techniques be combined to achieve indoor localization based on a map?

1.3. Contributions

The work in this thesis contributes to the field of indoor localization for robotic swarms using audio-based signals. These contributions are listed below:

- Combination of PF and SSL techniques in a novel approach using probabilities to achieve high accuracy indoor localization that works anywhere without training or environmental adjustments.
- Unlike similar works, only a building map is needed to perform indoor localization.
- Leveraging the unique characteristic of sound messages to increase the speed and accuracy of robotic localization.
- The design of a robotic car platform and the design of the software running on it in combination with a microphone array.
- Adjusting the audio codec to allow it to be used for the real-time sending and receiving of messages.
- Loading a map of a building from a .json file and generating multiple geometrical properties about it, like the shortest distance, longest distance, and path between cells.

1.4. Document structure

The rest of this document is structured in the following way. First, the related works of all separate techniques, combined in this thesis, are discussed in chapter 2. This is followed by some background information about the techniques used in this thesis in chapter 3. Next, an overview of the hardware and software written for this thesis is given in chapter 4. Then, the algorithm of this thesis will be described in chapter 5. This is followed by an extensive evaluation of the implemented system in chapter 6, where the results of the proposed method are displayed, and a comparison is given to other similar works. Lastly, chapter 7 gives the conclusion of the work in this thesis and a final answer to the problem statement.

2

Related work

This chapter discusses the current state-of-the-art localization techniques using other types of sensors and the different approaches to audio-based localization. Next to this, a small survey into the related works regarding the particle filter is given in this chapter.

2.1. Localization techniques

Indoor localization, more specifically indoor localization for Unmanned Vehicles (UVs), has been a popular topic of research that will only become more popular in the near future [47]. The need for robots that can localize themselves indoors and thus work without the user's input is becoming increasingly important in our society [48]. Over the years, numerous techniques and sensors were used to achieve this localization, each with its own advantages and disadvantages. The most popular sensors that are used are described below, where a short description of the technique is given, followed by a comparison to the method proposed in this thesis. To conclude, a summation of each technique's main advantages and disadvantages is given in Table 2.1.

2.1.1. Wi-Fi

Indoor localization using Wi-Fi signals was first proposed in [49] back in the year 2000. This paper describes localizing a mobile user indoors using three base stations (computers), where UDP packets are sent periodically from the user to the base stations. Using the signal strength at each base station that is extracted from the UDP packet, the user's location can be determined in a predefined area with an accuracy of around 2 to 3 meters. This paper and many other papers about indoor Wi-Fi localization use a technique called **fingerprinting** to determine the user's location. Fingerprinting consists of two phases: the *offline* phase and the *online* phase. During the offline phase, a data set is created by recording the signal strength at different angles throughout the building, resulting in a *radio map*. During the online phase, the same measurement is made again, and a technique like weighted k-nearest neighbors is used in combination with the data from the offline phase, which results in an estimated location of the user. A simple overview of the fingerprinting process can also be seen below in Figure 2.1. In contrast to the base stations, almost all the new work on Wi-Fi localization uses Wi-Fi routers and their received signal strength indicator (RSSI), which are available in almost every building at the time of writing. This approach significantly cuts down the deployment cost of the technique and makes it applicable to most public buildings. Guang-Zhong Yang et al. [12] proposes a variation that removes even more deployment overhead by removing the need for the offline phase. Here the direction of arrival (DOA) is derived from the RSSI of numerous access points, thereby eliminating the need for the offline phase. Next to this, there are many more variations of indoor Wi-Fi localization, all of which largely boil down to the same thing. In the end, sound-based localization was chosen over Wi-Fi-based localization because it doesn't travel through walls and removes the need to have stationary beacons inside the room, allowing it to work anywhere.

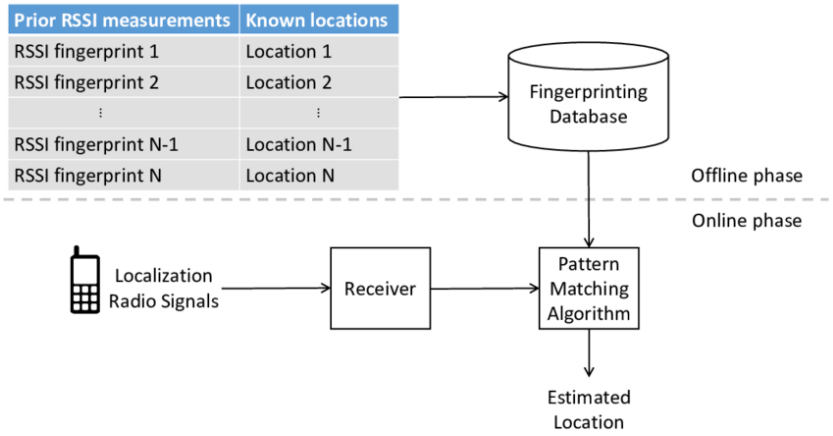
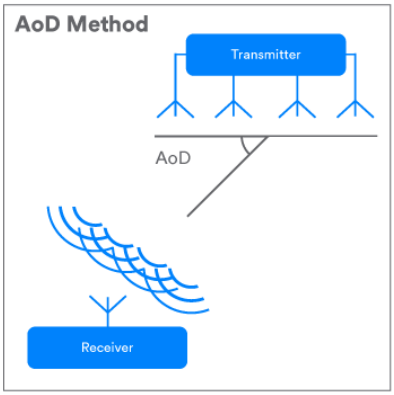


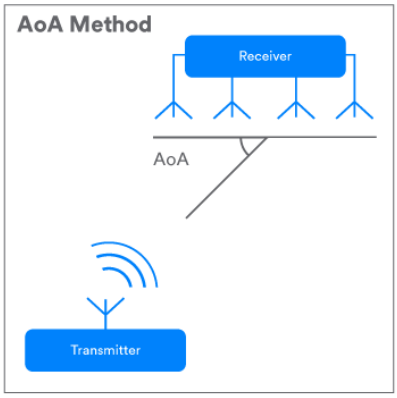
Figure 2.1: Fingerprinting algorithm [50].

2.1.2. Bluetooth Low Energy

Another popular indoor localization technique is Bluetooth, specifically Bluetooth Low Energy (BLE), a technique found on nearly all mobile devices nowadays [51]. BLE, just like Wi-Fi, is a technique that makes use of radio waves in the 2.4GHz band (2400 to 2483.5 MHz) [52], with the main difference between these two techniques being that Wi-Fi has a higher throughput and power consumption in comparison to a very low power consumption coupled with a lower throughput for BLE. Earlier attempts at indoor localization using BLE were based on using fingerprinting [53]. This, however, changed after the introduction of Bluetooth 5.1 [54] back in 2019, which introduced the Bluetooth direction-finding system. This new addition to the Bluetooth standard contains the ability to determine the Angle of Departure (AoD) and Angle of Arrival (AoA) using a multi-antenna array [55]. These techniques can be seen in Figure 2.2a and Figure 2.2b respectively. Using these techniques, the relative location of a user to a beacon (or other person) can be determined, allowing for easier and more efficient indoor localization.



(a) Angle of Departure (AoD).



(b) Angle of Arrival (AoA).

Figure 2.2: Bluetooth 5.1 directional capabilities.

The AoA and AoD are calculated by looking at the distance between the antennas in the array and the phase shift between the antennas with respect to the received signal. Using this information, trigonometry can be used to calculate the AoA and AoD between the transmitter and receiver, where the receiver calculates the AoA, and the transmitter calculates the AoD. Both metrics could, in theory, be calculated by the receiver if the receiver somehow knows the physical dimensions of the transmitter's antenna

array. To prevent this newly added functionality to the Bluetooth standard from interfering with the data transmission, a signal called the Constant Tone Extension (CTE), with a constant wavelength and frequency, is added to the back of a Bluetooth package. Rinaldi et al. [13] uses these new features of BLE to achieve an average AoA error of 5° for distance up to 4m, between an anchor and a BLE device, which can be used to calculate their relative position to each other. The main disadvantage, just like with Wi-Fi, compared to using sound-based localization, is that BLE signals also travel relatively easily through walls. This means that the robots will not be able to use this information to effectively localize themselves in an indoor environment.

2.1.3. Ultra-Wideband

Ultra-wideband (UWB), just like the techniques mentioned above, is a technique that also makes use of radio waves. It operates at the frequency spectrum of 3.1GHz to 10.6GHz [56] with channels of 500MHz. Because the channels are rather wide, the FCC applied strict power regulations for UWB. Because of this, UWB consumes only a small amount of power, making it an ideal technology for devices with a very limited battery capacity, like smart tags. Another big advantage of having such a wide channel bandwidth is that it makes UWB more immune to flaws as *multipath fading, interference,* and it has *improved timing* results [57]. Next to this, UWB also offers a better range than Bluetooth [58], and it can perform localization more accurately than when using Bluetooth or Wi-Fi [57].

One of the more frequently used techniques for localization using UWB is Time-of-Flight (ToF), which comes in two flavors. First, there is One-way ranging (OWR), which requires clock synchronization between the nodes, and Two-way ranging (TWR), which does not require any clock synchronization and is therefore the preferred approach. When using TWR, the transmitter sends out a message to a receiver who, after receiving the message, sends a message back containing the time it took from receiving the message to sending the message. Upon receiving this message, the transmitter can compute the relative distance to this node from the timing details it has gathered. Optionally, the transmitter can also, at this point, send another message (containing the processing time) to the receiver, allowing the receiver to obtain the relative distance to the transmitter. Applying this method using at least three anchor nodes around an object allows this object to be localized using ToF [14]. Another popular approach in UWB localization is Time Difference of Arrival (TDOA), which uses multiple clocks synchronized anchors to determine the location of an object by looking at the different arrival times of the signal at the different anchors. Morón et al. [15] propose a method that combines ToF and TDOA to achieve even higher accuracy in relative localization. Because most of the localization techniques of UWB are based on fixed anchors, the technique is unsuitable for this thesis's goal. Next to this, UWB also has poor scalability because timing channels need to stay open for every channel, increasing the processing power exponentially with the number of robots.

2.1.4. ZigBee

ZigBee is an open communication protocol developed by the ZigBee alliance in 2004 [59]. ZigBee works with radio waves at 68MHz, 916MHz, and 2.4GHz, achieving data rates of 20kbps, 40kbps, and 250kbps, respectively [60]. Because ZigBee has such low data rates, it can operate at very low power, especially when devices sleep between transactions, making it an excellent communication protocol for battery-powered short-range devices that need to run for years. Next to this, the ZigBee protocol is a self-organized multi-hop mesh network that supports up to *254 nodes* [60], allowing for excellent scalability and redundancy. A typical ZigBee network, as seen below in Figure 2.3, can be (automatically) configured in multiple topologies containing three types of devices. First off, there is the *ZigBee coordinator*, a Full Function Device (FFD) that acts as a PAN coordinator in the network, managing the structure of the network. This coordinator works with a specific Pan ID, which allows multiple networks to be used simultaneously next to each other. Secondly, there are *ZigBee routers*, which are also FFD and could potentially, in the case when the coordinator gets lost, act as a new coordinator in the network, making the network extremely redundant to node failures. Last up, there are *ZigBee devices*, these devices are also called Reduced Function Devices (RFD), meaning that they can only talk to FFD in the network and cannot operate without them. These devices are extremely low-power devices used for sensors or actuators in the network [61].

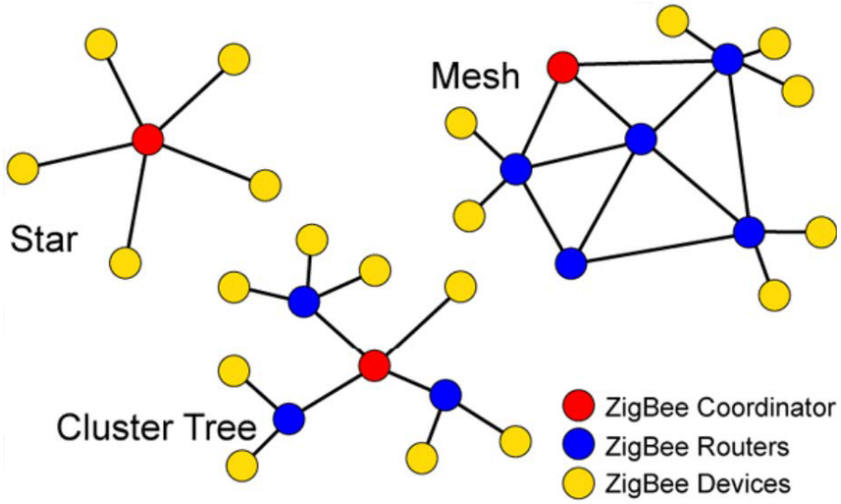


Figure 2.3: ZigBee network topologies.

Hai et al. [16] and Latine et al. [17] propose a ZigBee method to achieve indoor localization. The novelty in these state-of-the-art papers is that a third axis is added to the localization, allowing for the calculation of the elevation of the device of interest. Both methods work with a series of fixed anchors inside a building connected to a centralized processing unit. The object's location is calculated using triangulation by looking at the difference in the RSSI values of each anchor. Further research shows that not much more research, that uses a completely different approach, was done in recent years using ZigBee for localization. Because this method again uses fixed anchors, it is unsuitable for the goals of this thesis, namely achieving indoor localization without modifying the infrastructure of a building. Next to this, the use of centralized processing units in these works voids the property of a robotic swarm, where only relative communication is used.

2.1.5. RFID

Another radio waves-based localization technique is Radio-Frequency Identification (RFID). RFID operates mainly at three different frequencies: *Low frequency (LF)* (30 - 500kHz), *High frequency (HF)* (10-15MHz), and *Ultra high frequency (UHF)* (850 - 950MHz, 2.4 - 2.5GHz, 5.8GHz) [62], where it holds that the higher the frequency the higher the range. The RFID systems work with so-called readers and tags, where the reader sends out a 'read' request upon which the tag responds by sending back its unique ID contained in the chip [63], as can be seen below in Figure 2.4. The RFID tags can be categorized into two categories: *passive tags*, which do not have a power source and harvest energy from the reader's signal. And *active tags*, which do have their own power source and send out their ID at a specific interval. Having its own power source allows active tags to be used over long distances.

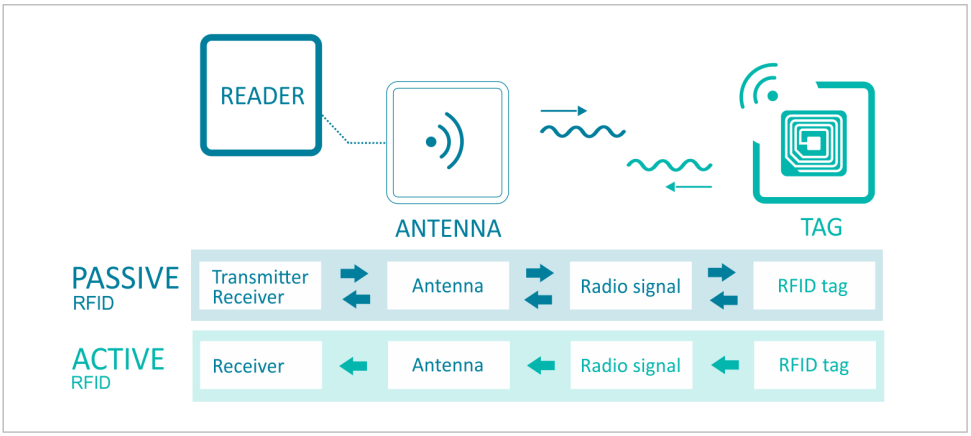


Figure 2.4: RFID workflow.

DiGiampaolo et al. [18] propose a localization method that uses UHF RFID tags containing three closely spaced antennas spread throughout a building. Having three antennas instead of one allows the reader to not only retrieve the distance to the tag but also its relative rotation towards the tag, allowing for better localization results. Bernardini et al. [19] uses rotating antennas and a synthetic-aperture-radar (SAR) to perform the localization, allowing for even better results. Nearly all localization methods using RFID require tags to be placed throughout a building, thus requiring existing infrastructure to be altered. Because of this, using RFID is not suitable for the goal of this thesis, as the goal is to develop an algorithm that can work anywhere without changing the infrastructure.

2.1.6. Light-based

Localization based on light is a popular method of localization because it involves something that is, in most cases, already available: light. De Silva et al. [21] and Celikkanat [20] propose a relative localization method using infrared (IR) light, allowing for non-invasive localization since humans cannot see light in that spectrum. Here, both papers use an infrared LED array in combination with an array of infrared receivers. The biggest advantage of using this technique is that it can simultaneously be used for obstacle avoidance and localization. When one robot sends out an infrared pulse, it can use the returned waves from this pulse to determine if there are any obstacles in its path. At the same time, another robot could receive this signal and use it to determine its relative position to the robot. A Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol is used to prevent signal collisions. At the same time, infrared light can be used for communication, even up to a point where audio can be streamed using an array of infrared LEDs [22].

Next to invisible infrared light, there is also the area of Visible Light Communication (VLC), where normal light sources (like LEDs) are used to perform the localization. Soner et al. [23] propose a method for localization that uses a car's headlights to determine their relative position towards each other and, simultaneously, allow them to communicate. Here, the cars communicate their relative position to each other to prevent a possible collision between the two from happening. Another approach could be to use an image sensor to boost the data up to 15Mb/s [24] and allow for real-time response to the environment. Besides an image sensor, a camera could also be used with an array consisting of RGB LEDs, which can use different colors and blinking patterns to transmit data, allowing for a high throughput combined with localization [25]. The one downfall of using light to achieve indoor localization is that it is very prone to interference from ambient light, requiring a high degree of processing power to filter this out in real-time, making it unsuitable for low-power robotic vehicles.

2.1.7. Lidar

Lastly, there is a special type of light-based localization that works with laser lights, called Lidar. Here, a laser is spinning fast, and based on the reflected light that is received back, a 3D point cloud of the environment is created [64]. Using this point cloud, a rough map can be constructed of the environment. Lidar is most used as an extra tool, in combination with one of the techniques mentioned above, when it comes to localization. Feng et al. [65], for example, propose a method to improve the accuracy of UWB localization by utilizing Lidar, improving the average localization error up to 0.10m. Xin et al. [26] and Li et al. [27], however, both propose a method that uses Lidar as its main localization sensor. Both papers use the Simultaneous Localization and Mapping (SLAM) algorithm, where a robot can construct a map of the environment while simultaneously determining its position on this map [66]. This is also the biggest advantage of using Lidar, as it works anywhere and always without needing a pre-designed map. While using Lidar is great for self-localization, it does, however, offer low accuracy when it's being used as a standalone method. Next to this, Lidar sensors are, at the time of writing, still very expensive, costing at least €100. This makes deploying it to all robots in a swarm costly, making this approach impractical for broad deployment.

	Advantage	Disadvantage
Wi-Fi	Infrastructure widely available.	Environment changes affect results.
BLE	Consumes only a small amount of power.	Inaccurate at larger distances.
UWB	Great accuracy and interference immunity.	Potentially less power efficient than BLE.
ZigBee	Self-restoring network.	Extremely low data rate
Light-based	Very low cost.	Influenced by ambient light.
RFID	Tags require no power.	Requires pre-installed readers or tags.
Lidar	Create a map of unknown rooms.	Low accuracy as a standalone method.

Table 2.1: Localization techniques main advantage & disadvantage.

2.2. Sound source localization

Sound Source Localization (SSL) is the ability to locate a sound source by only using sound [67] and is comprised of two parts, namely the distance and the direction of arrival (DOA). In contrast to the methods above, which use electromagnetic waves, sound is produced by vibrating the molecules in the air. Because of this, sound cannot traverse through solid objects as easily as most electromagnetic signals can. Next to this, sound has the property that it can (unlike light) bend around walls, making it possible for robots to find other robots by following the sound path through a door into the room of interest. Numerous attempts have been made to perform sound source localization, where most efforts have gone into localizing other sounds in the environment rather than other robots in a swarm. All these algorithms could also be applied to most of the techniques described in section 2.1. This section focuses mainly on localizing other robots, as this is the goal of this thesis.

2.2.1. TDOA

One of the earlier works of SSL is based on a technique that is still one of the most popular, namely *Time Difference of Arrival (TDOA)*. Dating back to as early as 1948, where Jeffress [68] implements TDOA in combination with two simulated ears to perform SSL to determine the position of a sound source in a room. TDOA is a technique that leverages the spatial distance between multiple microphones in a microphone array to determine the *Direction of Arrival (DOA)* and, in some cases, even the distance of a sound source [69]. A visual representation of this technique can be seen in Figure 2.5, where the sound reaches each of the N microphones at a different time. These time differences are determined by applying techniques like cross-correlation, generalized cross-correlation (GCC), or generalized cross-correlation phase transform (GCC_PHAT) [70–76] over the received data at each microphone. This results in detecting the exact same message at slightly different time offsets for each microphone. The TDOA that can be calculated from this is shown as d_1, d_2, d_n in Figure 2.5 and can be used to calculate the angle θ by applying geometry.

There are a lot of different ways in which TDOA is applied, for example with time-synchronized sound beacons [70, 71, 73] with a pre-determined location. By placing multiple of these beacons inside a room, the position of a system inside this room can be determined by sending audio from these beacons to that system. When looking at works that focus on localizing a robot/user, most of the works are only able to determine the DOA [75–77]. There are, however, also a few works out there that attempt to also calculate the distance to a sound source by levering geometry and using the TDOA information [69, 72–74, 78]. All these works distinguish themselves by the number of microphones used and how they are ordered in the array. Here, configurations that use from two [70, 71] up to an arbitrary number of N microphones [69, 72–79] in a single array are used. Next to this, how these microphones are shaped also affects the system's performance. There are configurations that use a straight line [79], square [71, 78], circular [75, 76], or triangle [73, 74, 77, 78] shape for the microphones. All these different configurations result in different performance results, where more microphones almost always result in a better performance at a higher processing cost.

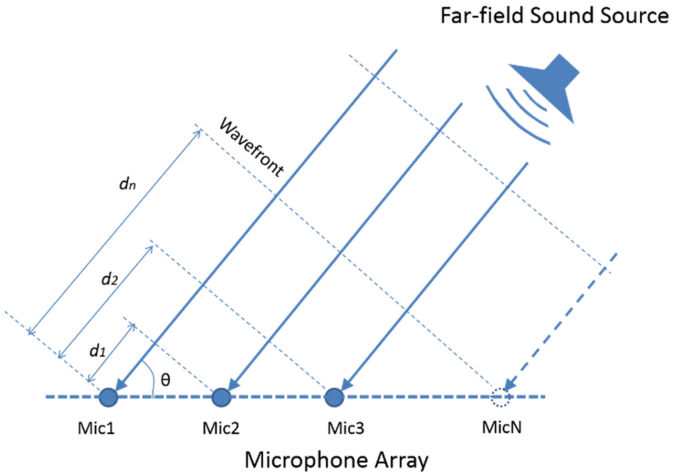


Figure 2.5: Basic TDOA principle [79].

2.2.2. MUSIC

The multiple signal classification (MUSIC) algorithm was originally developed for other types of wireless signals to determine the DOA of multiple received signals. This is done by exploiting the orthogonality of the eigenvectors' noise part and the signal's steered power response [80]. The whole MUSIC algorithm is performed inside the frequency domain, and it is mostly used for narrowband sound signals, even though there are also some attempts to make the MUSIC algorithm work for broadband signals [81]. The MUSIC algorithm comprises four main steps [82], starting with 1) computing the covariance matrix of the received signal. This is achieved by using Equation 2.1 and Equation 2.2, where A is the steered response matrix of the received signal, S is the source signal, and N is the harmonic noise vector for each microphone.

$$X = AS + N \tag{2.1}$$

$$R_x = XX^H \tag{2.2}$$

Next up, 2) performing eigenvalue decomposition, using Equation 2.3. Here, Q is the eigenvectors matrix, and Λ is the diagonal matrix containing the eigenvalues, where $\Lambda_i i = \lambda_i i$. Within this matrix the bigger eigenvalues are part of the actual signal, while the lower ones correspond to the noise that is present in the signal (N). These noise eigenvalues are orthogonal to any column of the steered response matrix A, allowing the retrieval of the DOA of the signal.

$$R_x = Q\Lambda Q^-1 \tag{2.3}$$

Then, 3) performing spatial spectrum estimation using Equation 2.4, which results in peaks in the directions where the sounds are coming from. Here E_n represents the signal vector, and $a(\theta)$ represents the direction candidate vector. This results in the last step being 4) selecting the peaks. Here the peaks of the outcome of step 3 are the DOAs of the different sound signals.

$$P_m u(\theta) = \frac{1}{a^H(\theta) E_n E_n^H a(\theta)} \tag{2.4}$$

Applying the MUSIC algorithm results in an accurate estimation of the DOA of multiple sound sources. However, this comes at a rather high computational cost. Because the goal of this thesis is to make the system work on a low-power device, and the fact that it is impossible to gain distance information, this technique was not considered feasible.

2.2.3. Beamforming

Beamforming is a filtering technique that uses steered power response (SRP) to steer the power of a microphone array to a specific angle [67], focusing on the actual signal that needs to be received and filtering out any noise coming from other directions. This process is also called spatial filtering, the process of splitting the noise from the actual signal by focusing on the DOA of the signal. The simplest beamforming implementation is the delay-and-sum (DAS) beamforming algorithm, which can be seen in Figure 2.6. This approach artificially shifts the signal coming from a specific direction to align the TDOA of the signal at the different microphones. This will result in a summed output with a high power (top part figure) representing the actual signal of interest, while the noise component (bottom part figure) will have a low power as its TDOA is not aligned. The output of such a DAS beamformer steered at a specific angle θ is given by Equation 2.5, where x_n is the received signal at microphone n, N is the number of microphones, t is the time, and τ_n is the TDOA between microphone n and the reference microphone at the steered angle θ. The energy of the beamformer at a specific angle θ can be calculated using Equation 2.6, which can be used to determine the initial direction of the sound source.

$$\widehat{S}_\theta[t] = \sum_{n=1}^{N} x_n[t - \tau_n(\theta)] \tag{2.5}$$

$$E_\theta = \sum_{t=1}^{T} \widehat{S}_\theta[t]^2 \tag{2.6}$$

The big advantage of this technique is that is very good at receiving multiple sound signals from different directions. However, the peaks in the steered direction are usually quite wide, which results in a poor resolution. This problem can partially be mitigated by adding whiting to the signal and processing it in the frequency domain. This technique is called steered-response power phase transform (SRP-PHAT) [83–85], which results in narrower peaks and, therefore, a higher resolution. This comes, however, at a computational cost, as the technique is quite computationally expensive, making it unusable for this thesis. There are some attempts made to make it less computationally expensive [86, 87], but it still cannot compete with the TDOA approach.

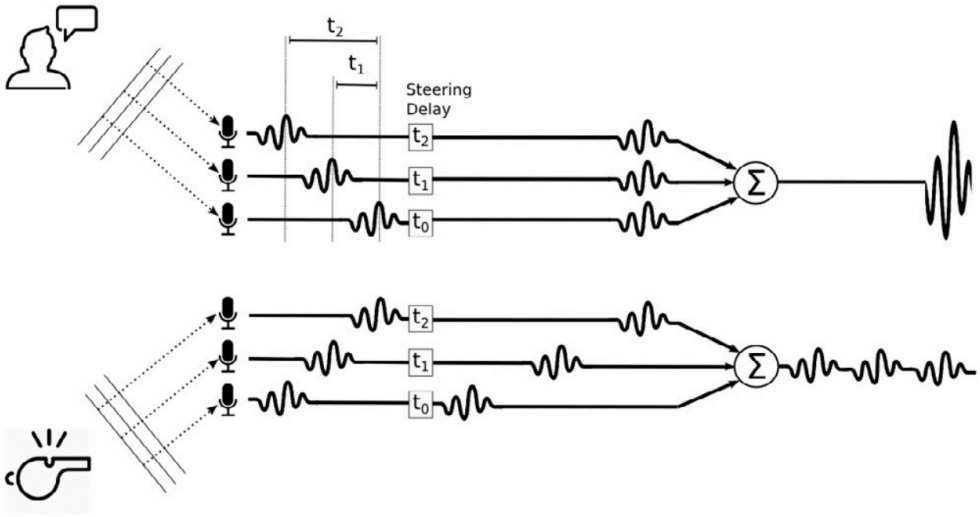


Figure 2.6: Beamforming delay-and-sum (DAS) method [67].

2.2.4. Machine learning based SSL

Since recent years machine learning has become a more and more popular area of research [88], having more than doubled since 2017. This is likely because the processing power and the development of machine learning have significantly increased over the past few years. Machine learning has also been widely researched for performing SSL, with vastly superior localization performance compared to the abovementioned traditional methods. The neural networks that are used in these works either accept features, like GCC or SRP-PHAT [37, 38] as the input or they take the whole audio signal as the input, whether it be in the time-domain [39] or the frequency domain [40]. Within the SSL machine learning domain, most work can be categorized into two categories, namely grid-free and grid-based SSL [41]. Feng et al. [42], Castellini et al. [43], and Zhai et al. [44] all apply such a grid-free SSL method, resulting in the relative angle (°) and distance (cm) to the sound source. Ma et al. [45] propose a grid-based machine learning SSL, where a convolutional neural network (CNN) detects a sound source on a 10x10 grid. Once the network is fully trained, this approach achieves near perfect and fast localization results. Mjaid et al. [28] uses a deep neural network (DNN) approach called AudioLocNet. In this work robots can communicate among each other using orthogonal chirp-based audio messages, these same messages are used to train the network. The output of this network is the relative location in a grid, up to 250cm around the robot, to the other communicating robot, as can be seen in Figure 2.7. This network is trained under three scenarios: Line-of-sight (LOS), non-line-of-sight (NLOS), and reverberant (Reverb) using thousands of recordings. This all results in accurate localization, even under noisy conditions (SNR < -10dB). However, the error metric used in this work is expressed in the number of wrong hops, which makes it difficult to compare it to other works. This state-of-the-art work is closest to the work performed in this thesis. Lastly, in [41], both grid-based and grid-free results are considered by utilizing a fully convolutional neural network (FCN) combined with a microphone array consisting of sixty microphones to estimate the location of multiple sound sources.

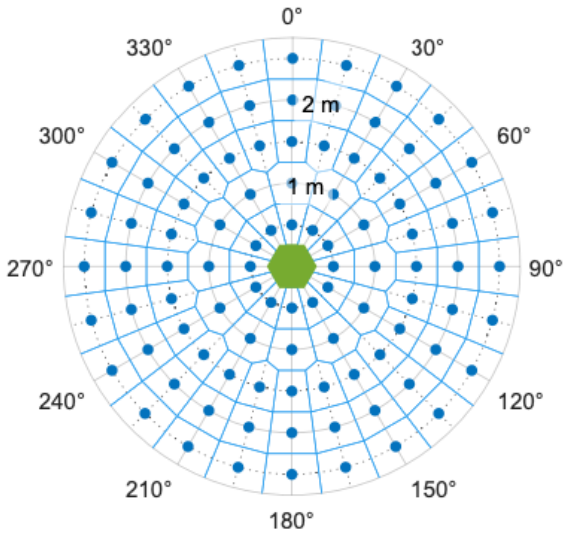


Figure 2.7: AudioLocNet localization grid [28].

All these approaches that use machine learning to achieve SSL use different types of layers, and, just as with the other SSL methods, they all use different microphone array configurations, each with its own advantages and disadvantages. The one thing that all works have in common, is that they outperform the standard approaches of SSL in the environment they were trained in. This is also immediately the biggest disadvantage of using machine learning, as it won't work in other environments before it is trained there. To make it even worse, simply changing a small thing in the trained environment, like the position of a plant, can negatively impact the performance of this method as that would change the room's acoustic response [89]. Next to this, most works are trained in relatively noiseless environments, which means that just like changing the environment, it will not work anymore once more noise is added (other than trained noise). Training the system for every location and noise level is also a tedious job,

and any changes in the environment would require new training data. Because of this, this thesis aims to achieve a performance similar to that of the machine learning-based approach but without machine learning, making the system usable in every environment.

2.3. Particle Filter

The particle filter algorithm is a probabilistic algorithm that uses a recursive implementation of Monte Carlo based statistical signal processing methods [90–92]. A great definition of the Monte Carlo method was given by Halton in 1970, namely: ”representing the solution of a problem as a parameter of a hypothetical population, and using a random sequence of numbers to construct a sample of the population, from which statistical estimates of the parameter can be obtained” [93]. This means that the particle filter algorithm calculates relevant probability distributions by using concepts of probability distributions with discrete random measurements and importance sampling. A detailed description of how the particle filter algorithm is executed in this thesis work can be found in section 3.3.

Previous works combining a particle filter with robotic swarms and SSL have proven quite effective. Valin et al. [94] and Otsuka et al. [95], for example, make use of a particle filter in combination with SSL to keep track of a talker sound source by employing particles with weights at the possible positions of the talker, in a video conferencing and moving talker robotic settings. Here, an original guess of the talker's position is made using original SSL techniques like steered beamforming, after which the location is tracked using particles. The addition of the particle filter is especially useful in mitigating errors that originate in reverberant environments, as false talker locations will have a lower particle weight and are, therefore, easier to recognize. This combination of techniques can achieve performances of up to a 1°error in angle and a 10% error in distance. Another approach of combining a particle filter with SSL is given by Wu et al. [96], where a particle filter is used with a robotic car swarm. In this work, the particles can interact with each other throughout the swarm, leading to improved convergence results. Zhang et al. [97], Ramazan et al. [98], and Zhao et al. [99] also make use of a particle filter within a robotic swarm, in combination with the Particle Swarm Optimization (PSO) [100] algorithm to achieve high accuracy (indoor) localization results. Lastly, Xu et al. [29] proposes a method that lies closest to the work that is done in this thesis, namely combining chirp-based signals with a map-based particle filter to achieve indoor localization. The work in this thesis, however, differs since it does not require the use of Bluetooth beacons, making it more versatile.

While a particle filter can achieve high accuracy localization results as a standalone algorithm, it might occur that in large non-unique environments convergence can take a lot of time or in the worst case might not even occur. This leads to situations in which the robotic car would possibly drive in the wrong direction for a long time before it can start navigating to the target location, making it not ideal for time-sensitive situations. Next to this, even though the fact the approaches mentioned above all come with great (indoor) localization results, they still all have some issues. First up, some of these approaches use some sort of beacons to perform localization, making the approach only usable in environments that are pre-decorated with these beacons. Secondly, many papers that combine particle filters with SSL have done that in the context of video conferencing, which is completely different from self-localization and is, therefore, not applicable to this thesis. Lastly, not all papers have used proper evaluation techniques, where a lot of the evaluation is only done for line-of-sight situations. This is insufficient, since the robot would also have to deal with reverberant and non-line-of-sight scenarios in a real-world scenario, making the evaluation incomplete.

3

Background

This chapter is intended to provide insight into the various techniques and principles that form the basis of this project. For each of these techniques, an explanation of how it works is given, followed by an explanation of how and why it is implemented in the proposed work.

3.1. Sound Source Localization

As mentioned before in section 2.2, sound source localization is the ability to locate a sound source by only using sound [67]. The output of SSL can be divided into two parts: the direction of arrival (DOA) and the distance relative to the source's position. These two parts will be treated below in subsection 3.1.1 and subsection 3.1.2 respectively. SSL takes the data from the microphones as its direct input and transforms that into a localization result, an overview of this process can be seen in Figure 3.1. The first step in SSL is *feature extraction*, where the data from the microphone array is converted into features. Here, there are different types of features, like the time difference of arrival (TDOA), where the time difference of the same signal at the different microphones is taken [101]. Another popular approach is taking the inter-microphone intensity difference (IID), where the energy difference at a point in time between the different microphones is compared [67]. After extracting the features from the microphone data, the features can be mapped into a localization result based on the used propagation model. For example, this can be done by a grid search, where the features of every possible location are recorded beforehand and used for comparison with the received features. Next to this, there are also different propagation models to consider based on the use case of the SSL. Free field propagation can be used when the sound from the source can reach the microphone array without any obstacles. Near-field propagation can be used when the sound source is close to the microphone array. Lastly, far-field propagation can be used when the sound source is far away from the microphone array, such that the inter-microphone distance and the distance to the sound source are such that the sound wave can be considered planar.

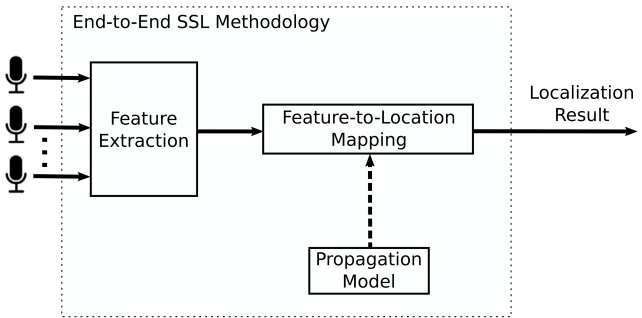


Figure 3.1: Data pipeline end-to-end SSL methodology [67].

3.1.1. Direction of Arrival

The first feature that will be extracted from the sound signal, to determine the robot's location, is the direction of arrival (DOA). The direction of arrival represents the direction, in degrees, from which a message is received from another robot. In other words, it indicates the direction from which the sound traveled from the other robot and, in the case of line-of-sight, the actual direction to the other robot. This DOA is relative to the robot's own orientation, where a DOA of 0 degrees means right in front of the robot. Here, the front of the robot is defined by the placement of microphone 1 on the microphone array, and it is indicated by the red arrow in Figure 4.5. With the help of the DOA, the algorithm, described in section 5.1, can eliminate multiple impossible locations on the map, narrowing down the list of possible locations for the robot. The feature extraction method based on the TDOA has been chosen to determine the DOA. Here, the arrival times at each of the six microphones are found using convolution to detect the preamble part of the message, as described in subsection 4.2.5. Using these arrival times, the TDOA can be calculated using Equation 3.1, where τ_{ij} is the TDOA between microphone i and j and t is the arrival time of the message at a specific microphone.

$$\tau_{ij} = t_i - t_j \tag{3.1}$$

This formula calculates the TDOA between each consecutive microphone pair. Next to this, the TDOA between opposite microphones is calculated to increase the accuracy and robustness of the DOA calculation by possibly correcting the angle pair by 180°based on the direction of the signal. Next up, the speed of sound needs to be calculated using Equation 3.2, where c is the speed of sound in m/s and T_C is the room temperature in °C. At an average room temperature of 20°C, the speed of sound using this formula would be 343 m/s.

$$c = 331 * \sqrt{1 + \frac{T_C}{273}} \tag{3.2}$$

The actual DOA can be calculated using the TDOA pairs and the speed of sound. Here, the first step is to calculate the angle pair for each microphone using Equation 3.3, where c is the speed of sound in m/s, l is the length between two consecutive microphones (**4.65cm**), and α is the angle between two consecutive microphones **(60°)**. The angle pair is optionally corrected here based on the direction of the signal, which is determined using the opposite TDOA values calculated in the previous step. Using the calculated angle pairs, the actual DOA can be calculated using Equation 3.4, where M is the number of microphones. This formula will output the direction in which the message was received as an angle between 0°and 359°.

$$AnglePair_i = \begin{cases} 180 - (\arcsin(\frac{\tau_{m_i,m_{i-1}} * c}{l}) * (\frac{180}{\pi}) - (i-1) * \alpha) & \text{if } \tau_{m_{i-1},m_{i-4}} > 0 \\ \arcsin(\frac{\tau_{m_i,m_{i-1}} * c}{l}) * (\frac{180}{\pi}) - (i-1) * \alpha & \text{otherwise} \end{cases} \tag{3.3}$$

$$DOA = 360 - \arg(\sum_{i=0}^{M} \exp(AnglePair_i * (\frac{\pi}{180}))) \mod 360 \tag{3.4}$$

3.1.2. Distance

The second characteristic feature for localizing the robot is determining the distance between two robots. As mentioned before in section 1.1, sound has the unique feature that it bends around walls instead of moving through them. Because of this, the actual distance traveled between two robots can be determined instead of the Euclidean distance. This, in combination with the DOA, allows for precise localization results using the algorithm of this thesis described in section 5.1. To achieve this, a technique is used that has been widely employed for determining the distance over RF signals and even sound signals [102–105], namely Round-Trip Time (RTT). Here, the distance between two robots can be calculated by looking at the time-of-flight of the signal in the air, which is proportional to the distance.

The general idea behind RTT is shown below in Figure 3.2. Here, robot A initiates the sequence by sending a localization request to the other robots while simultaneously storing the time of sending the message. When robot B receives this message, it immediately responds to robot A. When robot A receives this message, it will again store the time, from which the start time will be subtracted, resulting in the total time-of-flight of the signal between the two robots.

Because this approach of calculating the distance between two robots depends on the time that the signal spends in the air, it is important that accurate and predictable timing measurements can be obtained on the device that is running the algorithm. As can be read in section 4.2, most of the project runs on a Raspberry Pi, which has an operating system. Because of this operating system, time-sensitive tasks cannot be achieved properly since operating system tasks might interrupt the application, resulting in timing errors. This was also tested on a Raspberry Pi that was patched with a real-time kernel[1], which allows the kernel to be fully preemptable and should improve real-time performance. However, this was still not real-time enough and resulted in unexpected timing results. Because of this, this part of the project is performed on the microcontroller of the robotic car itself. Since this microcontroller does not require an operating system to work, it can achieve predictable and accurate time measurements. To transform the number of air-time samples found on robot A into the actual distance between robots A and B, the number of samples between sending a message at robot A and receiving a message back from robot B is recorded while sampling at *44.1kHz*, which is the maximum supported sampling frequency for the microphones.

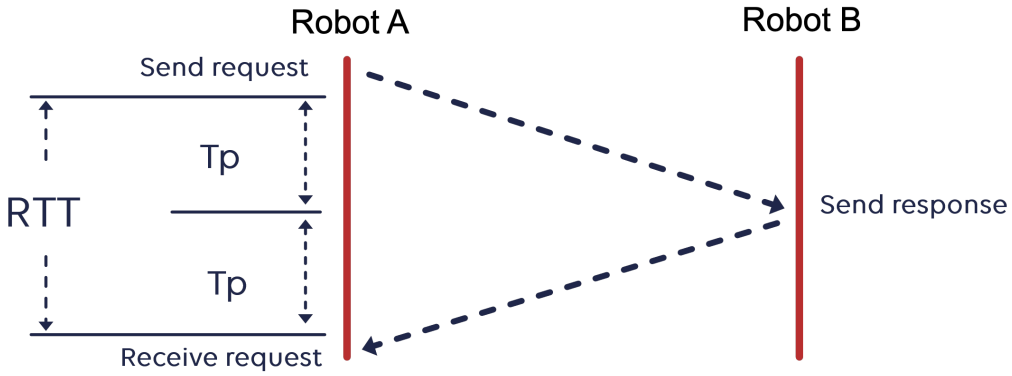


Figure 3.2: Round-trip time (RTT) between robots A and B.

The first thing that is needed is the speed of sound, which can be calculated using Equation 3.2. Next, the number of samples in a one-way delay (sending a message from one robot to another) must be derived from the total number of recorded samples. This is done using Equation 3.5, where OWD is the number of samples in a one-way trip and $RTT_{samples}$ is the total amount of samples found between sending and receiving back on robot A. P_B is the processing time in number of samples on robot B, which is constant since this part is executed on a microcontroller and it was measured to be *670 samples*. Lastly, D_B is the delay taken, in the number of samples, at robot B before sending back a response to robot A. Here it was chosen to give every robot a unique delay, based on its ID, that is known by all robots inside the swarm. This allows all robots that can hear the distance request from robot A, to respond to it without interfering with each other.

$$OWD = \frac{RTT_{samples}}{2} - P_B - D_B \tag{3.5}$$

Using the speed of sound found using Equation 3.2 and the one-way delay (OWD) found using Equation 3.5, the actual distance between two robots can be calculated using Equation 3.6. Here the OWD is transformed into the actual airtime by dividing it by the sampling frequency F_s, which in this case is

[1] https://wiki.linuxfoundation.org/realtime/start

44.1kHz. Then the airtime is multiplied by the speed of sound to get the distance between the robots in meters. Lastly, the result is multiplied by 100 to get the distance between two robots in cm.

$$Distance = \frac{OWD * c}{F_s} * 100 \tag{3.6}$$

3.2. Chirp signals

For the communication between the robots inside the swarm, as well as performing the sound source localization, audio-based signals are used in this thesis because of the reasons described in section 1.1. Encoding data over audio can be achieved by modulating the audio signal using a pre-defined modulation schema. One approach would be to use *narrowband* signals to modulate the signal, however, because of the small bandwidth used with this technique it suffers from poor data rates and constructive and destructive interference [106], leading to a poor overall performance. Another approach would be to use *wideband* signals, which, due to their bigger bandwidth, have a larger data rate and are also less susceptible to both forms of interference. However, such signals on their own are hard to detect under noisy conditions, as they tend to blend in with the noise. Because of this, chirp-like signals were chosen for this thesis, as they offer impeccable noise resilience and good correlation properties.

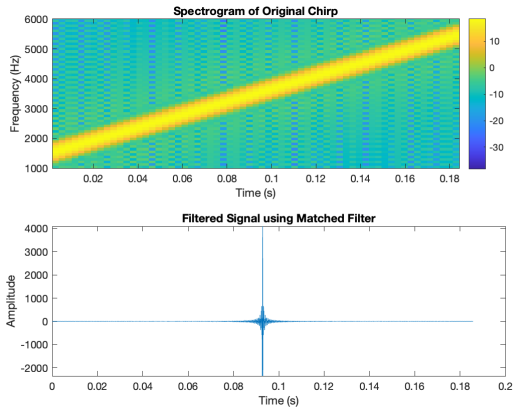


Figure 3.3: Preamble linear chirp.

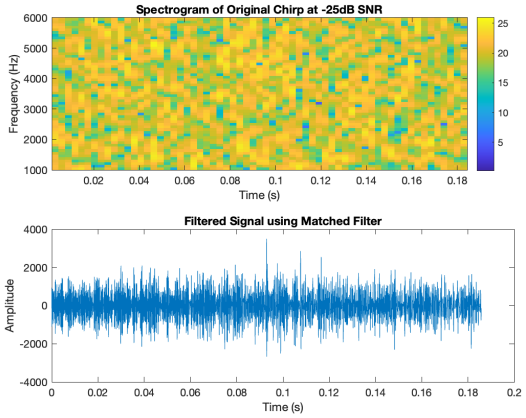


Figure 3.4: Preamble chirp at -25dB SNR.

As seen in Figure 3.3, a linear chirp is a signal that performs a linear frequency sweep from one frequency to another during a specific time. This figure displays the chirp used in the preamble, which sweeps between 1.5kHz and 5.5kHz during a time of 0.18 seconds, which results in *8192 samples* at a sampling frequency of 44.1kHz. Because of the unique nature of such signals, they can be easily distinguished from noise, as seen in Figure 3.4. This figure shows that a clear peak, representing the presence of a preamble, can still be seen in the matched filter plot, even under extremely noisy conditions at -25dB SNR. Figure 3.5 also shows the great correlation properties of a chirp signal, where the preamble chirp can clearly be distinguished from white Gaussian noise and the data chirps, allowing for overlapping transmissions between robots. What can also be noticed here is that the signal's amplitude decreases at the beginning and end of the chirp signal. This is because a Kaiser window is applied over the signal, which is further explained in subsection 4.2.4. Lastly, a chirp signal is also very robust against the Doppler effect, as this will only result in a small error in delay because Doppler is shifting the frequency, and the frequency is linearly modulated throughout the waveform [107]. This means a preamble can still be correctly detected, even when the cars are moving during transmission. The only disadvantage to this error in delay is that it will impact the distance measurement, but this is out of this project's scope.

These chirp-like signals are generated by first calculating the frequency at a specific time, using Equation 3.7. Here, f_t is the frequency at time t, f_{start} and f_{stop} are the start and stop frequencies of the chirp respectively, and d is the duration of the signal in seconds. For these signals, the value of t will always be in the range $[0, x]$, where x is determined by the number of samples of the chirp and the sampling

frequency f_s. As can be read in subsection 4.2.4, this would be *8192 samples* for the preamble and *768 samples* for the other data. Using this calculated frequency, the actual signal can be calculated using Equation 3.8, where s(t) is the signal at time t, a is the signal's amplitude between 0.0 and 1.0, and t is the time.

$$f_t = f_{start} + \frac{(f_{stop} - f_{start}) * t}{2 * d} \tag{3.7}$$

$$s(t) = a * cos(2 * \pi * f_t * t) \tag{3.8}$$

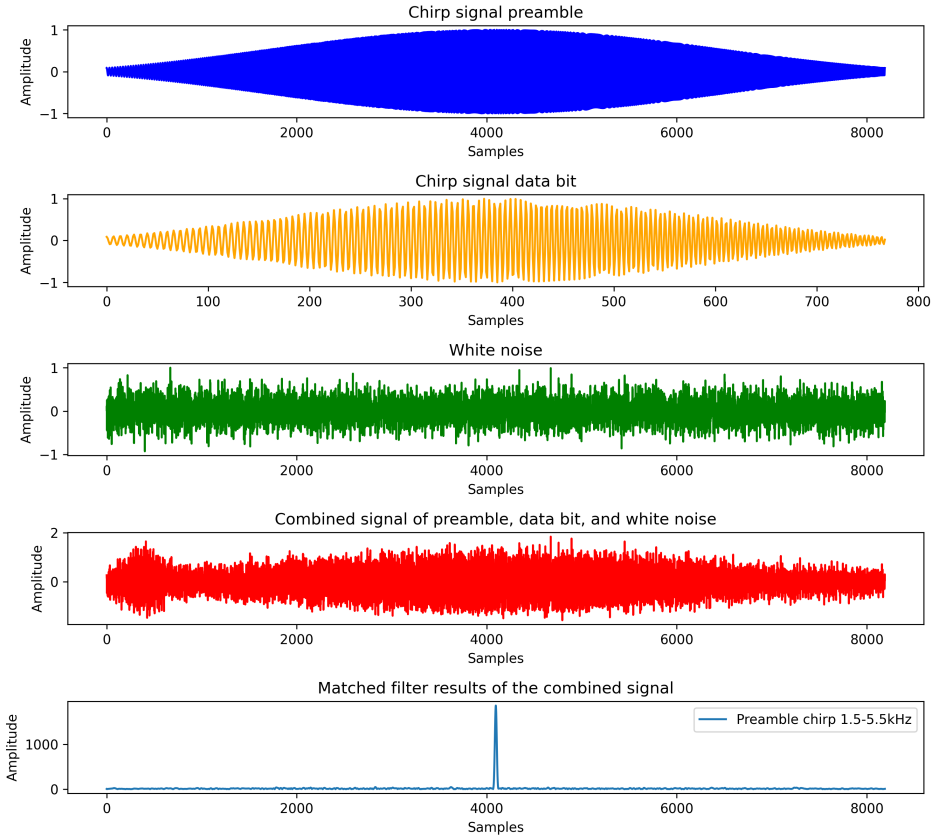


Figure 3.5: Correlation properties chirp.

3.3. Particle Filter

This project combines a conventional particle filter with sound source localization to achieve improved indoor localization results. This means that besides the proposed algorithm, the robot's movement will also update the particle filter. This will increase the accuracy of the overall system since, upon movement, any errors made by the algorithm can be corrected, and the confidence for a location can be increased. The particle filter used in this project uses a map of the building, which is divided into several cells as explained in subsection 4.2.6. On this map, a total of 10.108 particles are placed, each representing the possible position of the robot with a size of 1x1cm. This number of particles was chosen as it has proven to offer great localization results while not being incredibly computationally demanding and it allowed each cell to have an equal number of particles at the start. An overview of the implemented particle filter can be found in Figure 3.6, where the flowchart of the particle filter is depicted. The first step in the particle filter algorithm is to *initialize the particles uniformly*, where all the 10.108 particles will be divided uniformly over the map. Here, each cell will get an equal number of

particles assigned to it. The weight of each of these particles will be initialized according to Equation 3.9, where P_w is the particle's weight, and N is the total number of particles. This means that the robot could be in any of the cells at this stage with equal probability. The particles can also be drawn onto the map, which updates the position of the particles in real time. The uniformly spread particles can be seen in Figure 3.7. The weight of the particles represents their importance and lifetime during the algorithm, as the longer the particle exists, the higher its weight becomes, and thus, the higher the chance becomes that this particle will represent the robot's actual position.

$$P_w = \frac{1}{N} \tag{3.9}$$

After the initialization of the particle filter, the algorithm will start updating the position of the particles when movement is detected. This movement comes from the sensors on the robotic car, where the IMU is used to find the robot's relative angle with respect to north and the hall sensors inside the motors are used to find out when a full rotation of the wheels has been made, as this is when the particle filter will be triggered to update. Since the diameter of the robot's wheels is known in advance, the distance traveled by one rotation can be calculated using Equation 3.10, where d_w is the distance traveled by one wheel rotation in cm, n is the number of rotations, and d is the diameter of the wheel in cm. Using the angle and distance, the particle filter will calculate the movement of the robot along the x and y axis using Equation 3.11 and Equation 3.12, where m_x and m_y are the movements along the x and y axis in cm respectively, d_w is the distance of one wheel rotation, and θ is the angle in which the robot has traveled.

$$d_w = n * d * \pi = 1 * 12.5 * \pi = 39.27cm \tag{3.10}$$

$$m_x = d_w * sin(\frac{\theta * \pi}{180}) \tag{3.11}$$

$$m_y = -d_w * cos(\frac{\theta * \pi}{180}) \tag{3.12}$$

Next up, the algorithm will start looping over all particles on the map and calculate their new possible locations by using Equation 3.13 and Equation 3.14, where x and y are the original coordinates of the particles, m_x and m_y are the movements along the x and y-axis, and e_1 and e_2 is the added *Gaussian noise* for the x and y coordinate respectively. This noise is calculated separately for both the x and y coordinates using Equation 3.15, where e is the noise in cm and $\mathcal{N}(0, 1)$ is the normal distribution used to select a random noise which is constrained between $\pm 5cm$ to prevent the particles from moving too far from the target position. This noise is added to each particle to account for any sensor noise or errors in the motion model. Next to this, the noise also helps prevent particles from getting clustered onto one point, and it helps to maintain diversity among the particles, helping to explore a broader state space [108, 109].

$$x_{new} = x + m_x + e_1 \tag{3.13}$$

$$y_{new} = y + m_y + e_2 \tag{3.14}$$

$$e = \mathcal{N}(0, 1) \tag{3.15}$$

Using the possible new location of the particle, the next thing done is to check if these coordinates are valid. This is done by checking if the coordinates lie within one of the cells and if the path traveled from

the original position to the new position does not traverse any of the walls on the map. If the particle's new coordinates are valid, its coordinates are updated to these new coordinates, and its weight is increased by adding the weight calculated by Equation 3.9. However, if the particle's coordinates are invalid, the particle is marked as invalid and stored for resampling. After processing all particles, the next thing that needs to be done is *resampling all incorrect particles*. During this step, all incorrect particles their weight will be reset to the original weight, and they will be placed close by one of the correct particles, where some Gaussian noise is added to the particle's position. The particle to which they will be placed is chosen using an empirical distribution, where the particles with a higher weight have a greater chance of being chosen. After resampling, the weight of all particles will be normalized again, such that the total sum of weights is again one. Lastly, using the new locations of all particles, it is checked if the robot can already be localized. For the robot to be localized, at least 70% of all the particles must be in one cell, or the deviation between all particles must be lower than a certain threshold (400cm).

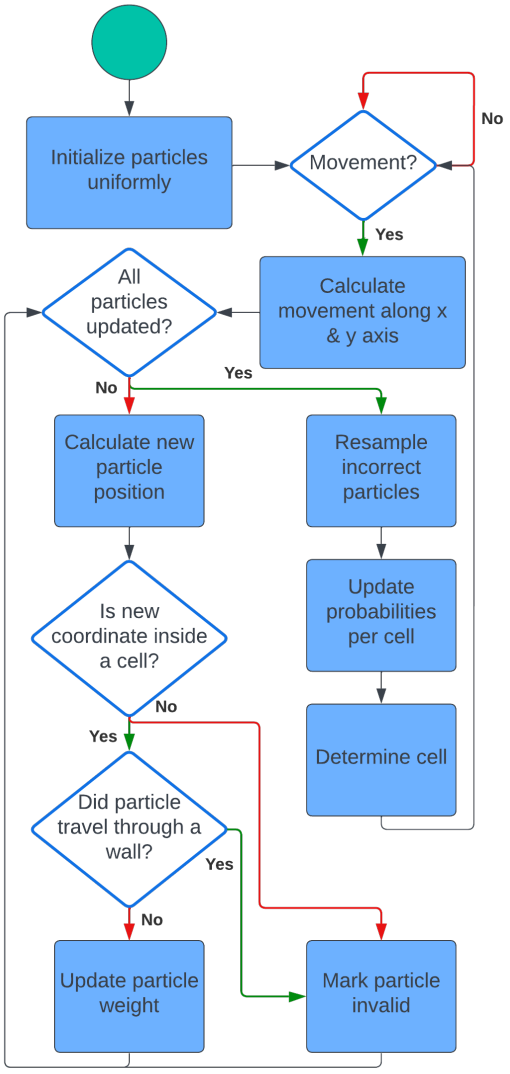


Figure 3.6: Flowchart particle filter.

Figure 3.7: Map of the top floor of my building, filled with particles.

4

System Overview

This chapter will give a complete overview of the implemented system of this thesis. First, the hardware used for this thesis project is described. This is followed by a description of how the software was implemented and why some of the choices were made during the project's development.

4.1. Hardware

The hardware used for this thesis project can be split into two parts. First up, the robotic car platform consists of a microcontroller board equipped with numerous sensors and four motors. This part is responsible for maneuvering the car and reading the onboard sensors. Secondly, there is the microphone array board that is connected to a Raspberry Pi, which oversees the communication and running of the algorithm. These two parts are connected to each other via a serial connection, which the Raspberry Pi can use to transmit driving instructions to the car, and the car can use it to relay sensor information back to the Raspberry Pi.

4.1.1. Robotic car

The first piece of hardware used for this thesis project is the robotic car platform, shown below in Figure 4.1 and Figure 4.2 without the microphone array connected and in Figure 4.3 and Figure 4.4 with the microphone array connected. This car has a microcontroller board that features an STM32F767ZI microcontroller that is running at 216MHz and is connected to four DC motors. These four motors can individually be controlled to allow the robot to move in any direction. Since the motors are connected to this board, this part of the hardware is solely responsible for maneuvering the robot, reading out the robot's sensors, and determining the distance between two robots as mentioned in subsection 3.1.2.

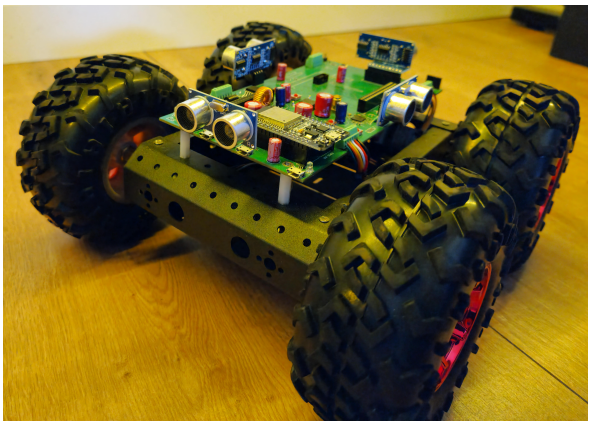


Figure 4.1: Front view robotic car w/o microphone array.

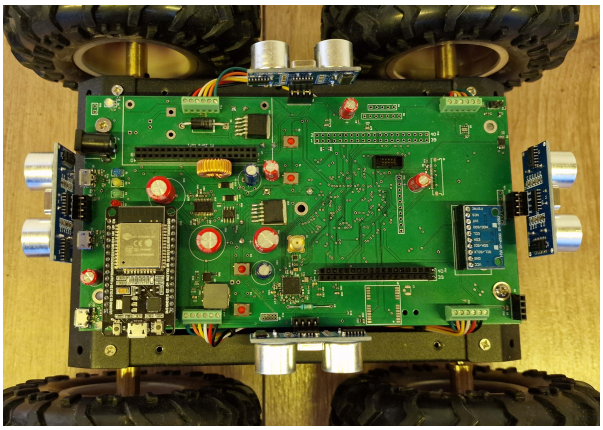


Figure 4.2: Top-down view robotic car w/o microphone array.

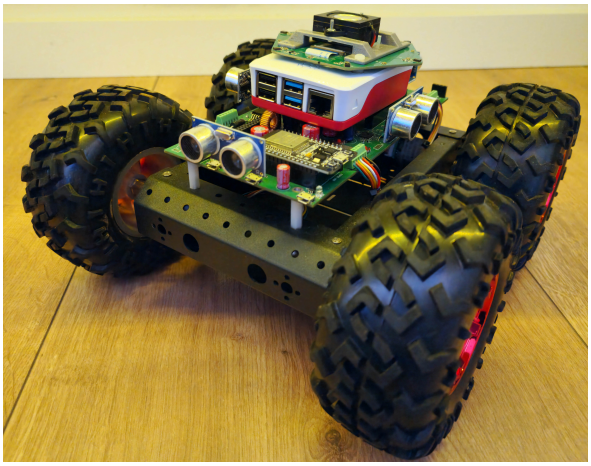


Figure 4.3: Front view robotic car with microphone array.

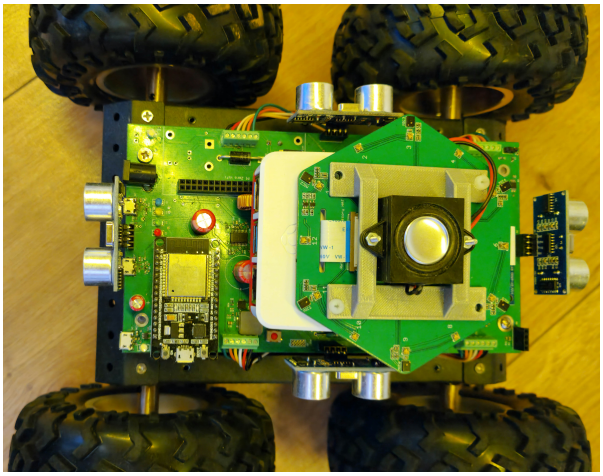


Figure 4.4: Top-down view robotic car with microphone array.

Because the board is equipped with numerous sensors and actuators, it makes it the ideal platform for robotic car development and research. This is also why this board was chosen for this thesis project, as it offered the freedom to attach any kind of hardware easily and without any problems. Next to this, it offered a good separation between the maneuvering of the car and the proposed algorithm in this thesis. For this project, the boards for the cars were completely made from scratch, meaning that everything on the board is hand-soldered, and the software for the board is written from scratch. In total, six cars were completed for this project, forming a swarm of robotic cars together. Here, each car has a size of 30x28cm. In terms of software, only a basic implementation was made, consisting of a communication protocol, motor control, simple obstacle detection using ultrasonic sensors, gyroscope measurements from the IMU, and reading the temperature in the room. Next to this, the four motors are equipped with hall sensors, allowing for synchronization between the four wheels and the detection of a full wheel rotation, which can then be relayed back to the particle filter algorithm as described in section 3.3. Besides all these features, the robotic car platform has even more connections and sensors that were not needed for this project. A full list of the specifications of this board can be found below in Table 4.1.

Main microcontroller	STM32F767ZI	
Clock speed	216 MHz	
Secondary microcontroller	CC1352P	Used for wireless communications like ZigBee, Lora, etc.
Motor driver	DRV8908	
Motors	DT25-310	12v, 300rpm
Maximum speed	1.9635 m/s	
Ultrasonic sensors	4	HC-SR04
Size	30x28cm	
Other features	Temperature & humidity sensor GPS Inertial Measurement Unit (IMU) Camera connectivity Four status LED's On-board amplifier Temperature & humidity sensor Lidar connectivity	 SkyTraq S1216F8-GL GY-9250 blue, yellow, green, red PAM8302AASCR HDC1080DMBR
Connectors	ESP32 Raspberry pi General purpose	 Used to connect microphone array.

Table 4.1: Robotic car specifications.

4.1.2. Microphone array

The second piece of hardware used in this project is the microphone array, which is used for all sound-related tasks, such as determining the DOA, distance, and communication. For the microphone array, the choice was made to use an off-the-self array made by Respeaker, namely the Respeaker 6 mic array kit [110]. This kit consists of a microphone array consisting of six microphones, which are connected to the two integrated ADCs on the board. These six microphones are evenly spaced in a circle, each with 4.65cm between them. Next to this, a 6 Ω 2W speaker from Grove [111] is used to output audio, which is connected to the integrated DAC of the board. This speaker is mounted precisely in the middle of the array using a bracket, ensuring that the sound emits from the center of the robot, allowing for optimal localization conditions. This microphone array is mounted directly to the 40-pin header of a Raspberry Pi 4B, which was possible since the microphone array is specifically designed for a Raspberry Pi. The microphone array can be seen below in Figure 4.5. The microphones used on the board are *MSM321A3729H9CP MEMS microphones*, which, according to the data sheet, have a frequency range of 100Hz-48kHz. This means that according to the Nyquist theorem, which states that the sampling frequency should be at least twice as high as the highest frequency in the signal [112], only frequencies of up to 24kHz can be used. After testing, however, it quickly became clear that the microphones do not perform well at their upper limit, where signals above 18kHz were not detected. Because of this, a lower sampling frequency of **44.1kHz** was chosen for this project, allowing the usage of signals up to 18kHz. The full specifications of the Respeaker board can be found in Table 4.2.

This microphone array was chosen for this project because of three main reasons. 1) Ease of development, since the microphone array consists of a ready-to-go kit that can easily be connected to a Raspberry Pi, and all the drivers are already available. This meant that the development of this thesis project could immediately start without wasting time and effort on designing and making a new microphone array and writing a driver for that. 2) The board was already available, meaning that no time and money had to be wasted by ordering other microphone array boards. 3) The board met this project's minimum requirements since it has audio input and output capabilities. One of the downfalls of this board is that it is specifically designed for a Raspberry Pi, which means that it only has a driver for the Raspberry Pi and can thus only work with a Raspberry Pi. During this project, it was tried to create a custom driver to make the board work on the microcontroller of the robotic car. This, however, turned out to be a very difficult task, as there is no documentation available on the board, and reverse engineering the Raspberry Pi driver would take too much time. Because of this, the choice was made to run parts of the project on a Raspberry Pi, to save on engineering work and focus more on the research part of the thesis. These parts will then communicate with the microcontroller on the robotic car and instruct it where it needs to move.

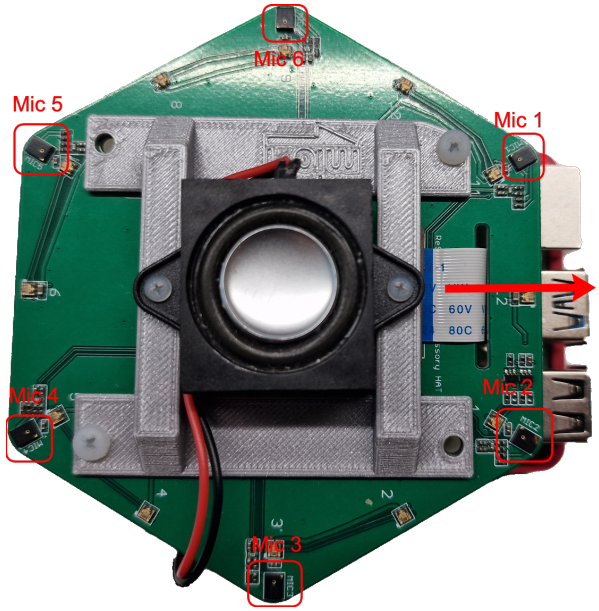


Figure 4.5: Microphone array top view.

Num. microphones	6
Microphones	MSM321A3729H9CP
Distance between mics	4.65cm
Angle between mics	60°
Frequency range	100Hz-24kHz
ADC	AC108
DAC	AC101
Speaker	Grove speaker plus

Table 4.2: Respeaker 6mic microphone array specifications.

4.2. Software

For this thesis project, quite a bit of software was developed on the Raspberry Pi and the robotic car. This section will, however, not include the software developed for the robotic car, as that was already mentioned shortly in subsection 4.1.1, and it's not the main part of the research. Also, the basic building blocks, like the DOA, distance, and particle filter, will not be mentioned in this section, as they have already been covered in chapter 3. As mentioned before, in subsection 4.1.2, an off-the-self microphone array is used for this project with a pre-written driver. When working with it, however, it was discovered that this driver has numerous faults, like the microphone order being different every time the application is restarted. So, this section starts by addressing these numerous issues and showing how they were solved. This is followed by the implementation of the messaging protocol and the initialization of the map.

4.2.1. Determining microphone order

The system comprises six physical microphones, as seen in Figure 4.5. However, the data received from the microphone array indicates that eight microphones are present. This can be explained by the fact that the board is equipped with two AC108 ADC chips, both 4-channel analog-to-digital converters. Because of this, two out of these eight channels will always be empty (read: have no signal data), making them useless for this project. Next to this, the order of the microphones is different every time the program opens a connection to the microphone array. To mitigate these issues and ensure the microphones are ordered correctly, a small algorithm was developed, as seen in Figure 4.6. A useful fact here is that the empty channels are always channels 7 and 8, followed by channels 1, 2, etc., making ordering the microphones easy once channel 8 has been identified. To find these two empty channels, first around 2000 samples are acquired. Secondly, the average deviation over all these samples is calculated using Equation 4.1, representing how much the signal deviates from 0 (center).

$$\sigma = \frac{1}{N}\sum_{i=0}^{N} |x_i - \bar{x}| \tag{4.1}$$

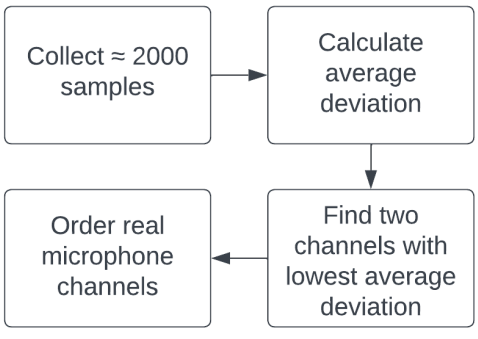


Figure 4.6: Flowchart microphone ordering.

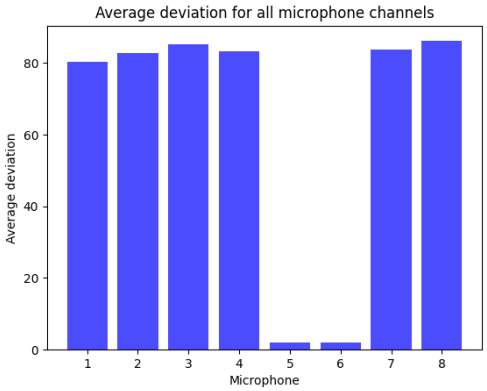


Figure 4.7: Average deviation for all microphones.

This results in the data that is shown in Figure 4.6, from which it immediately becomes clear that in this case channels 5 and 6 are the empty channels as these have a significantly lower average deviation than all other channels. After selecting the two channels with the lowest average deviation, the other six channels are ordered. After ordering the microphones, one can know for sure that the data from channel one in the software is the actual data from microphone one. Next to this, this step also removes the additional computational overhead of processing two more channels.

4.2.2. Calibrating the speaker output

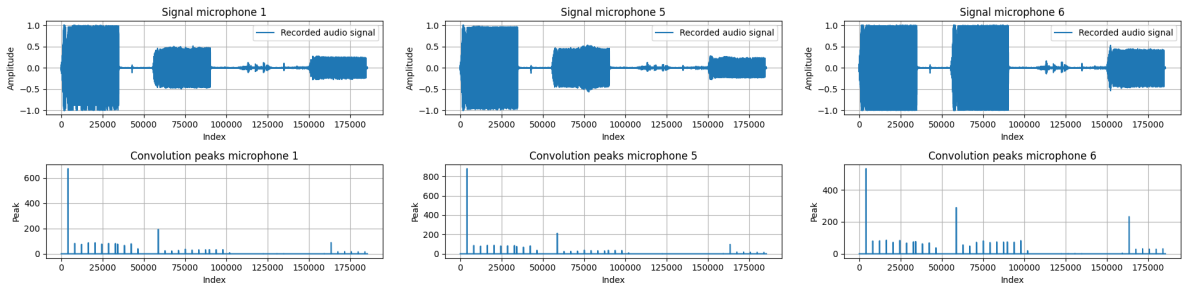
Another unfortunate downfall of using the Respeaker microphone board is that the volume output from the speaker is inconsistent between different runs of the application, where sometimes the volume is considerably (4-5 times) louder than other times. This leads to inconsistent decoding and filtering results, resulting in occasional performance degradation. To prevent this from happening, a calibration procedure is executed at the start of the application, which adjusts the volume gain of the application to get a consistent volume output. Here, the volume is directly related to the signal amplitude, which can be set between 0.0 and 1.0 and is initially 0.5. The calibration process is started by outputting the preamble, encoded at the current volume, *three times* and recording the average signal energy from its six microphones. The desired signal energy value is stored in the configuration file to make the output volume consistent across all robots. This has been done because this value is not consistent for every robot. Based on this recorded average signal energy, the decision is made to adjust the volume of the signal using Equation 4.2 and Equation 4.3, where D is the desired signal energy from the config, E is the currently detected signal energy, and $\triangle V$ is the amount of gain with which the volume will be adjusted. After this, the signal is re-encoded with the new volume, and the process is repeated until the recorded signal energy is in between the desired bounds.

$$\triangle V = sign * \frac{|D - E|}{4E} \tag{4.2}$$

$$sign = \begin{cases} 1 & \text{if } D - E > 0 \\ 0 & \text{if } D - E = 0 \\ -1 & \text{if } D - E < 0 \end{cases} \tag{4.3}$$

4.2.3. Filtering out own signal

The system can simultaneously send and receive messages. Because of this, a necessary step in the decoding process is filtering out messages originating from the same device to prevent any possible errors. As described in subsection 4.2.5, the maximum convolution peak is used to check if a preamble is detected in a specific window. This is also the same place where the filtering is applied since the convolution results of a message originating from the device itself are expected to be much bigger than that of a message from another robot. To verify this statement, various recordings were made for three different sources: the same device, a robot positioned right next to the recording device (10cm center-to-center), and another robot at 20cm distance. The results from these experiments can be seen in Figure 4.8 for microphones 1, 5, and 6, which were closest to the other robot positioned at 10 and 20cm. This figure makes it clear that a signal originating from the same device has a much higher convolution peak value than that of a signal originating from another robot, even if the two are close. To filter out the device's own signal, a threshold of **400** is set, meaning that all detected convolution peaks above *400* are considered messages from the same device and are therefore discarded. The value *400* was chosen after running multiple experiments and concluding that a message from the device itself always results in a convolution peak of more than *400* and a message from another robot always results in a convolution result lower than *400*.



(a) Convolution data microphone 1 **(b)** Convolution data microphone 5 **(c)** Convolution data microphone 6

Figure 4.8: Convolution data for own signal (1), signal at 10 cm (2), and a signal at 20 cm (3).

4.2.4. Message encoding

Communication, like determining the distance and the DOA, is achieved using audio-based signals. To achieve this, the data is modulated into an audio signal, which can then be sent to the other robots in the swarm using the speaker connected to the robot. Each created message has a fixed length of **80 bits** and a fixed order. The data frame of a single message can be seen below in Figure 4.9, and the encoded version can be seen in Figure 4.10. Every single data frame consists of the following elements:

- **Preamble:** The preamble is placed at the front of each message and it is used to indicate the start of a message. Next to this, the preamble is used to synchronize the message frame, indicating the exact position of the next part (Robot ID) of the message. This allows for the proper decoding of the message as it is known exactly where to expect the data of each bit. This part of the message is also used to determine the different arrival times of the message at each of the six microphones, which can be used to determine the DOA as described in subsection 3.1.1.
- **Robot ID:** This part of the message indicates which of the robots has sent the message. This also indicates the format for reading the bits in the rest of the message, as can be read below.
- **Message ID:** This part of the message is used to indicate the type of the message. This also indicates what kind of data can be expected in the data segment of the message. A complete overview of the supported message types can be read later in this subsection.
- **Data:** The data segment consists of **64 bits** and can contain different things depending on the message type.
- **CRC:** The last part of the message is the cyclic redundancy check (CRC), which is used to check if the message that was sent is still intact. For this, a simple XOR like CRC is used on all data, excluding the preamble. This type of CRC was chosen because of its simplicity and low computational complexity.

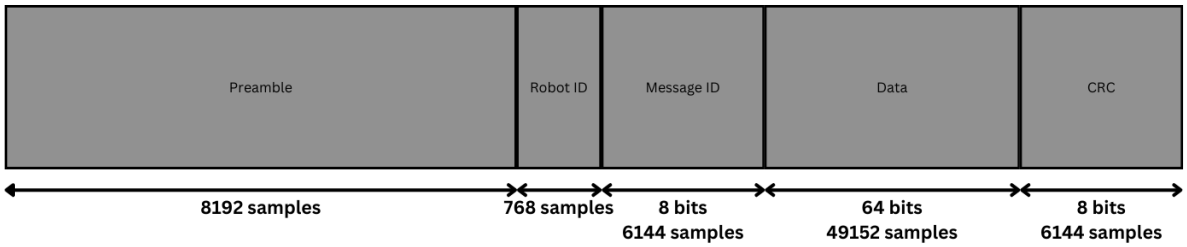


Figure 4.9: Data frame.

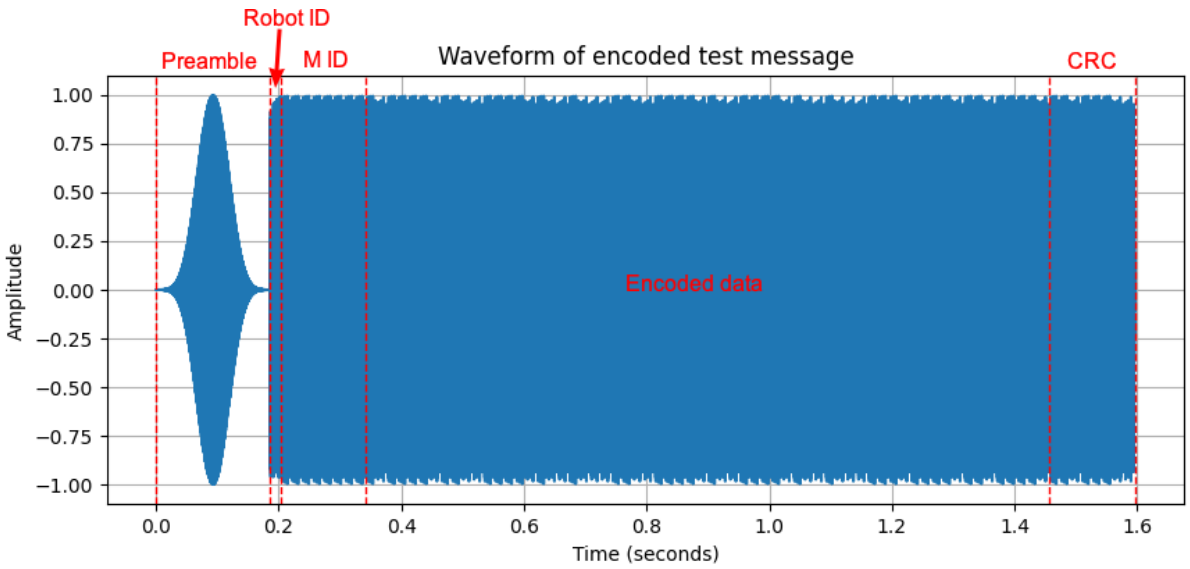


Figure 4.10: Encoded data frame.

When constructing a new message to be sent to one of the other robots in the swarm, the first thing that is done is generating the *preamble* and adding it to the front of the message. The preamble is a linear up-chirp from **1.5 - 5.5 kHz** during a time of **185.7ms (8192 samples)**, as can be seen in Figure 4.11. This number of samples was chosen after evaluating the performance of different sample sizes under noisy conditions, as can be read in section 6.4. Next to this, a comprehensive reason of why chirp signals are used to encode the message can be read in section 3.2. The next parts of the message are encoded using chirps within the frequency range of **5.5-18kHz**. The choice for this frequency range was made based on the supported maximum frequency of the microphone array, where the bandwidth for encoding the robot ID and the bits was maximized. Since the microphone array has a maximum sampling frequency of 48kHz, it was impossible to encode the message using non-audible sound signals. However, technically this should work when hardware is used that supports the encoding of messages at such high frequencies. The only change that would be needed, is changing the frequencies inside the configuration file.

As seen in Figure 4.9, the second part of the message is the *Robot ID*. For the encoding of the Robot ID, channel access mode Frequency-Division Multiple Access (FDMA) is used. This means that the available bandwidth of **12.5kHz** is split into equal parts of **2083 Hz** based on the total number of robots in the swarm (six), where each robot gets its own slice of the bandwidth. This approach allows multiple robots to communicate simultaneously, without interfering with each other, since each robot uses its own band to encode the robot ID. This allows the robot ID to be easily decoded, as described in sub-section 4.2.5, and upon finding the robot ID, the remaining bits can be decoded using the right patterns. As can be seen in Figure 4.12, the robot ID is encoded using a unique pattern of eight sub-chirps that each consist out of **96 samples** and has a total duration of **17.41ms (768 samples)**. This pattern is used by each robot using its own designated bandwidth and is unique with respect to the encoded preamble and bits, allowing it to be easily distinguished during multi-robot transmissions.

The remaining part of the message (Message ID, Data, and CRC) is encoded by first translating the data into a series of *80 bits*. These bits are encoded using a pattern that is based on the work of Mjaid et al. [28], where a pattern is used which is unique for each robot and for each bit. Based on the results of the evaluation of the messaging codec, which can be found in section 6.4, it can be seen that the results begin to deteriorate when more robots are added to the swarm. This can partly be explained by the fact that for each added robot, the frequency bandwidth needs to be divided by two more to create the unique patterns that allow simultaneous transmissions. Deploying six robots leaves around 1kHz of bandwidth for each sub-chirp, which, after testing, turned out to be the lower limit. Since the current hardware only allows frequencies up to 18kHz, allowing more robots into the swarm is impossible since

this would lower the bandwidth per sub-chirp below the 1kHz threshold. Because of this, it was chosen to use only six robots in this thesis work and leave the research into more robots as future work. Since the messaging codec supports six robots in the swarm, there are twelve unique patterns to be used to encode the bits. As can be seen below in Figure 4.13 and Figure 4.14, each bit is encoded using a unique pattern of twelve sub-chirps each with a length of **64 samples** and a total duration **17.41ms (768 samples)**. The key factor here is that each of these twelve patterns has low cross-correlation properties with each other, allowing for the concurrent transmissions of multiple robots. This also becomes clear when looking at the frequency plots in Figure 4.13 and Figure 4.14, where the patterns of an encoded 0 and 1 are completely different, allowing for easy distinguishing.

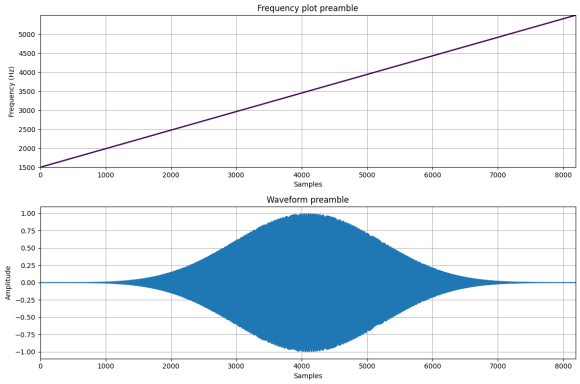


Figure 4.11: Encoded preamble.

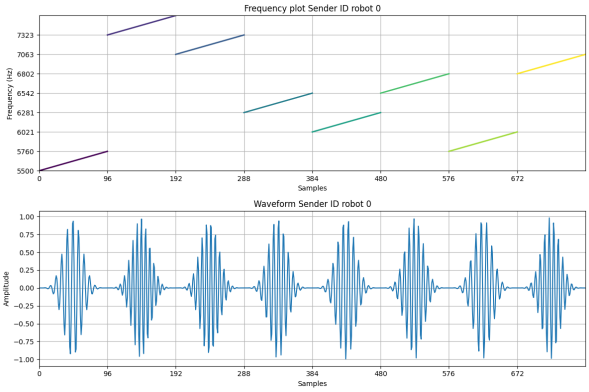


Figure 4.12: Encoded sender ID for robot 0.

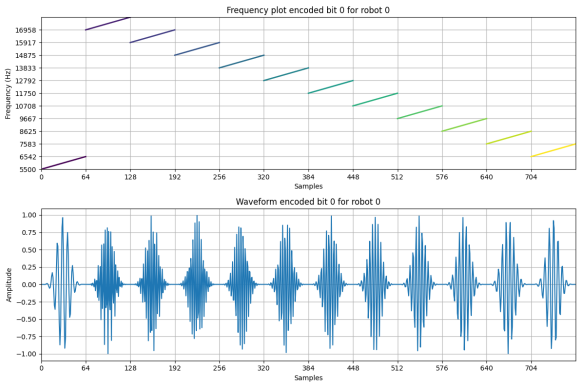


Figure 4.13: Encoded bit 0 for robot 0.

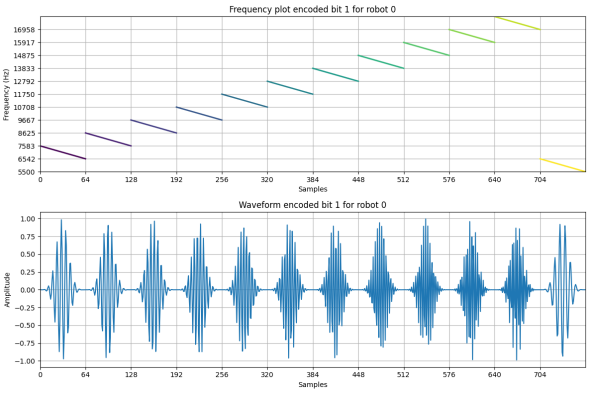


Figure 4.14: Encoded bit 1 for robot 0.

During the encoding of each (sub-)chirp, a **kaiser window** [113] with an $\beta = 14$ is applied in the time domain of the generated chirp signal. This window is applied to mitigate the effect of spectral leakage [114], therefore providing a more clear and accurate representation of the signal. This also focuses more on the signal's main lobe, reducing the side lobes of the signal. To achieve this, the windowing function reduces the amplitude of each (sub-)chirp at the beginning and the end of the signal, which can also clearly be seen in Figure 4.11, Figure 4.12, Figure 4.13, and Figure 4.14. Utilizing this Kaiser window also allows the encoding of all bits to be stitched together without the need for any padding around the bits since the beginning and the end of the signal are already reduced in strength, preventing leakage. The $\beta = 14$ value for the Kaiser window was chosen to minimize the interference from any side lobes and increase the robustness of the signal against noise. The final parameters used for the messaging codec can be found in Table 4.3. Here, some results are already depicted, which were derived during the evaluation described in section 6.4.

Sampling rate	44.1 kHz
Max nr. of robots	6
Kaiser window beta	14
Preamble frequency	1.5 - 5.5 kHz
Preamble samples	8192
Preamble duration	185.76ms
Bit frequency	5.5 - 18 kHz
Bit samples	768
Bit duration	17.41ms
Data rate	57.42 bps
Noise resilience (White Gaussian)	-12dB SNR

Table 4.3: Audio codec specifications.

Each message that is sent is encoded with a unique message ID, which indicates what kind of action needs to be executed by the receiver and what kind of data to expect in the data segment of the message. Not every message type has been fully implemented out of the list of supported message types. These message types have been added as proof-of-concept and could be utilized in future work to improve the proposed algorithm. The robots support the following message types:

- **Test:** This message type indicates a test message is sent. Here, the text: ”Hello!!!” is sent to another robot. This message type is used during the evaluation of both the DOA and the messaging codec, as it didn’t matter what the content of the message was.
- **Distance:** This message type is used by a robot to broadcast to all other robots that it would like to start the distance measurement. Upon receiving this type of message, each robot will respond to it within its own time interval as described in subsection 3.1.2.
- **Distance Response:** This message type is used by robots receiving a distance broadcast from another robot to send back a response. The original broadcaster of the distance message will, upon receiving such a message, calculate the distance to the responding robot.
- **Cell:** This message type indicates that the robot has localized itself in a certain cell, and the ID of this cell is embedded in the data segment of the message. This information could be used in future work to update the localization table for that specific robot and further increase the localization results.
- **Wall:** This message type indicates that the robot has detected a wall, and the angle at which the wall has been found is embedded in the data section. This information could be used in future work to update the localization table of a specific robot based on the geometrical properties of the map and increase the localization results.

4.2.5. Message decoding

The decoding of messages, upon receiving a message from another robot, is executed in real-time. This means that each bit, which represents a sample received by one of the six microphones, that is received, is immediately processed bit-for-bit. An overview of how these incoming bits are processed can be seen below in Figure 4.16, where the message decoding flowchart is depicted. Here, upon receiving a bit, the first thing done is to store this bit in the buffer corresponding to the microphone from which this bit was received. Since the microphone array consists of six microphones, six different buffers are also available to store the incoming bits. Next up, it is checked whether enough bits, which in this case is *8192 bits* corresponding to the size of the preamble, have been received. If this is the case, the program will take a frame out of the buffer of the last 8192 bits to determine if a preamble can be detected within this frame. This is done by firstly performing convolution over the received data and the flipped version of the original preamble, using the steps depicted in Figure 4.15. First off, FFT convolution is performed to determine how much the two signals overlap. Next, a *Hilbert transform* is performed over this data, followed by taking the absolute value of each item in the resulting data, which is done to get an envelope of the convolution data focusing on the extremes in the data.

After performing these three steps, the average and maximum value are determined from the resulting set, and it is checked whether the maximum peak is greater than at least *eight times* the average value. If this is the case, it is considered that a preamble is detected, and a decoding result object is created and stored in a list. Here, the maximum value needs to be at least eight times higher to prevent the error of a false positive while still being able to detect the preamble correctly, even under noisy conditions. As can be expected when looking at the convolution steps, performing this algorithm can become quite computationally expensive when the number of bits increases. Because of this, under-sampling was introduced to only perform the convolution over each fourth element of the current frame, reducing the computational load by four times, as described in section 6.4. Upon detecting the preamble, the detection position is saved in the decoding result object to indicate the position for which the rest of the message can be expected. Next to this, this preamble detection is executed for all six microphones, and upon detecting all six positions, the DOA can be calculated as described in subsection 3.1.1.

Figure 4.15: Convolution flowchart.

After successfully detecting the preamble, the robot will start to decode the robot ID of the sender and the remaining bits present in the message. This is done using the samples received by microphone 0, since doing the same for all six microphones is unnecessary. Next to this, not decoding the bits for all six microphones will save a significant amount of computational power. During this part of the code, which is always executed after the preamble detection, it will start to loop over the list of decoding results, and for each item, it will check if enough bits have been received with respect to the decoding position. Here, precisely 768 bits need to be received, since both the robot ID of the sender and the bits are encoded using 768 samples. If this is the case, the program will take a frame out of the buffer of the last 768 bits and use this to either decode the robot ID of the sender or the next bit. Here, it will always first decode the robot ID of the sender, since this information is needed to determine which pattern to use when decoding the bits. To decode the robot ID of the sender, the same convolution steps as depicted in Figure 4.15 are used to perform convolution with the flipped version of the originally encoded sender IDs of all robots in the swarm. Next, the convolution result with the highest peak is found, and it is assumed that this robot will have sent the message. This approach has proven to work effectively, with a success rate of more than 90% in the worst-case scenario, as can be read in section 6.4.

If the robot ID of the sender is already determined when taking the frame, the program will use the frame to decode the next bit of the message. Decoding a bit is again done using the same convolution steps as given in Figure 4.15, where convolution is performed with the flipped representations of the encoded bits for that specific robot ID as determined earlier. Here, the convolution result with the highest peak value is considered the truth, and either a 0 or a 1 will be added to the decoded bits list stored in the object. After decoding a bit, the program will check if all 80 bits have been decoded. If this is the case, the CRC will be calculated over the received bits and compared to the CRC present at the end of the message. If both CRC values are equal, the message has successfully arrived, and the result will be processed properly based on the message type. If this is not the case, the BER, in the case of a test message, will be calculated and printed to the console and after that the message will be discarded as the data cannot be trusted to be valid. Here, when receiving a message of the type *'Distance Response'*, the robot will still be able to use the gathered information about the DOA and distance to update the proposed algorithm of this thesis.

Receiving messages from multiple robots simultaneously is technically possible since each robot uses a unique pattern to encode its robot ID and the bits. Because of this, no overlapping frequencies should occur during transmission, which allows simultaneous transmission as long as both preambles do not completely overlap. However, implementing this functionality on the robots would require some form of Carrier-sense multiple access with collision avoidance (CSMA/CA) [115] to avoid the collision of multiple messages. Next to this, the order in which the messages would be processed needs to be

defined, which could impact the performance of the proposed algorithm. Because of this, it was chosen to consider multiple robot transmissions as out-of-scope for this work, and instead a fixed order was used where each robot communicates individually at its own time. This way, the results of the proposed algorithm would not be affected by any errors that might arise from multiple robots talking at the same time.

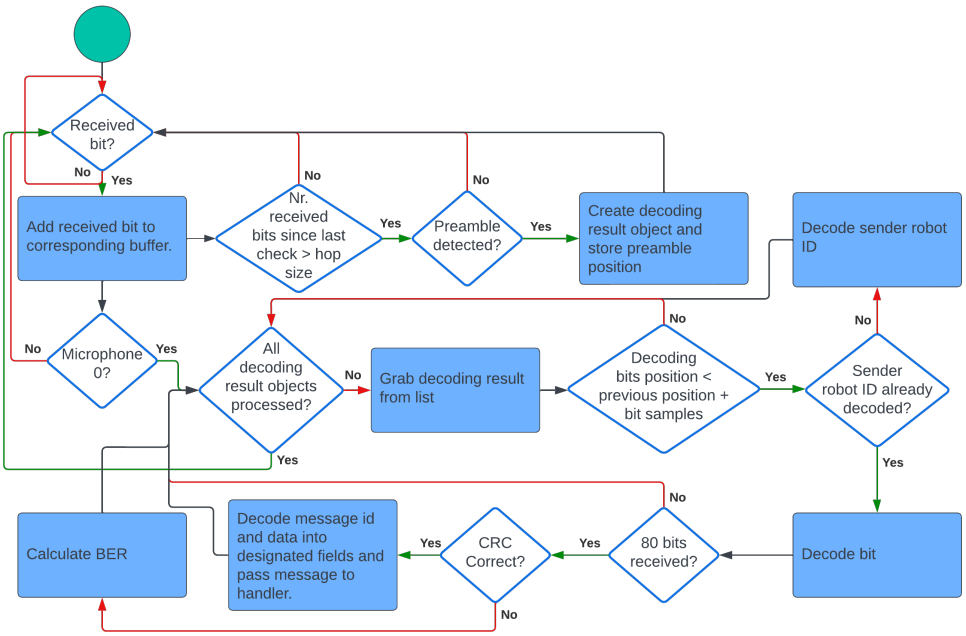


Figure 4.16: Message decoding flowchart.

4.2.6. Map initialization

For the proposed algorithm of this thesis to work, a map detailing the layout of the place where the robot will drive needs to be provided. This map data can be loaded into the program via a .json file containing a list of all wall coordinates, door coordinates, the map's name, and either a list of predefined cells or a set of allowed coordinates for the map. Here, each wall, door, cell, or allowed coordinate is described by a start and stop x and y coordinate, representing the real placement of these objects with respect to a pre-defined starting point. This means that the width and height of each element, which can be calculated by subtracting the stop from the start x/y coordinates, represents the actual width and height of that object in the room in cm. Using this information, the program can also draw this map to the screen, as seen in Figure 3.7 where the map of the top floor of my house is drawn. Here, it can also be seen that this map is subdivided into cells, each with a size of 40x40cm. These cells are used to localize the robot, meaning that when the robot is localized, it is inside the localized cell. This cell size was chosen based on the size of the robotic car (30x28cm), as the robot can completely fit into cells of this size, and the maximum error will be limited to **12cm**. Another reason this cell size was chosen is based on the average error found during the distance estimation evaluation, which can be seen in section 6.3. Here, the average error is around *20cm*, which means that the calculated distance is either 20cm too short or too long. This, in turn, means that localizing a robot in cells smaller than 40cm would not work, as the distance calculation algorithm is not accurate enough for that.

Next to this, the program also has a built-in feature to generate cells of a specific size when no specific cells are specified in the map's .json file. Here, the cells are generated by first taking the minimum and maximum x and y coordinates of the provided allowed coordinates. Then, the algorithm starts by trying to make a cell using the minimum x and y coordinates as the start coordinates of the cell. If the cell is not fully inside the allowed coordinates, the algorithm will update the start x coordinate and try again until a valid cell is found. When the cell is completely inside the allowed coordinates, the cell is

created, and the next cell will be tried in the same row at 40cm further, making the cells joining. When the algorithm reaches the maximum x coordinate, it will increase the start y coordinate by 40cm, and it repeats this process until the whole map is filled with cells. This basic algorithm, however, only fills the map with cells to a certain extent due to circumstances where a cell with its normal size will never fit and thus leaves open space. These special situations and how they are handled are described below:

- When filling a row with cells and reaching the right side of the map, where a cell with a width of 40cm might not fit anymore. Here, the cell's width is decreased so that it will exactly fit in the space left between the previous cell and the wall at that side. When this width is smaller than 5cm, the cell is not added, and this space is left open as localizing in such a small cell is not useful.
- There might be walls on the map that cause cells with their full height of 40cm not to fit anymore. In these situations, this row is treated as a separate row and it is filled with cells with a smaller height to fit the remaining space here. This is done in this row until the map again allows for cells with a bigger height, after which it will use the original cell's height again. Like the previous point, cells with a height lower than 5cm are not added, and the space is left open.
- Unique spaces in which a normal cell will not fit. The coordinates that are not filled with a cell, besides those left open by the previous two points, at the end of the program are filled with custom-created cells, filling up the remaining parts of the map.

Next to generating the cells, there is other data that needs to be calculated at the start of the program for the algorithm to work, namely the shortest and longest path between the cells and the path between the cells. This is only calculated once at the start of the program to save on computational costs while running the algorithm itself. The shortest and longest path between cells is calculated using the *A* algorithm* [116], which is a path-finding algorithm that is commonly used to find the shortest distance between two points on a grid using nodes of 10x10cm. This algorithm is optimal and efficient as it looks at the cost of each step, where it considers the cost to the target node. This cost is determined by looking at the start and end cost, where the start cost is calculated by summing up the number of nodes that are passed to reach the current node, and the end cost is determined by calculating the Euclidean distance to the end node, by using Equation 4.4. Here d(p, q) is the Euclidean distance between points p and q. For each new node it is checked if the node is valid by checking if the whole node is encompassed by one of the cells, if this is not the case the node is discarded, and this path is marked as invalid. The *A* algorithm* was chosen for this thesis as it was suitable to work on a microcontroller, and it allows for efficient calculation of the distances without using significant resources. A start and stop set of coordinates must be defined beforehand for this algorithm to work. This is done by defining a list of the coordinates of the cell's border, which are equally spaced from each other at 5cm. From this list of border coordinates either the two coordinates that are closest together or the two coordinates that are furthest apart from each other are chosen from both cells, for the shortest and longest distance respectively. This is determined by again using the *Euclidean distance* to calculate the distance between all border coordinates of both cells.

$$d(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{4.4}$$

When the *A* algorithm* reaches the end node, the algorithm is finished, and the distance can be calculated. This distance is calculated by referencing the parent node, which is available in each node. Here, at each step, the distance between the current node and the parent node is calculated and added to the sum of the final distance. During this step, it is also checked in which cell the current node is and whether it has been seen before. This is used to reconstruct the path (cells passed) between the two cells. This information is needed to accurately determine from which direction the sound can be expected when coming from a certain cell. After calculating all this data at the start of the program, the data is cached to a file, which can be loaded again the next time the program is run. This prevents the rather big calculations from needing to be performed every time the program is started and thereby speeding up the startup time of the program significantly. This cache file stores the generated cells, shortest distances, longest distances, and paths between the cells.

5

Algorithm

This chapter describes the algorithm of the proposed work of this thesis. Here, an extensive overview, describing how the algorithm works and all its different configurations, is given. Next to this, the choices that were made during the development of the algorithm will be elaborated in this chapter, giving a complete overview of the proposed work of this thesis.

5.1. Fusion of PF and SSL techniques

The main contribution of this thesis work, and the answer to the research question of this thesis, which can be found in section 1.2, is the fusion between the particle filter and the SSL techniques used in this project, described in earlier subsections, to achieve accurate indoor localization results. This novel approach to determining the location of a robotic car is achieved by updating the locations and weights of the particles based on the data received by the SSL algorithm. Here, a distinction can be made between three configurations: 1) The robot hears another robot, 2) The robot receives localization data from another robot about himself, 3) The robot receives localization data from another robot about another robot in the swarm, that is not himself or the sender robot. These three configurations have a lot of overlap when it comes to processing localization tables, but also a lot of differences when it comes to generating the localization table. Because of this, all three configurations are evaluated separately and will therefore be discussed separately below.

1) Hearing another robot: When the robot hears from another robot in the swarm, it will receive a list of probabilities depicting the probability that the robot sending the message is in a certain cell. This list is based on the particle distribution of that robot and can directly be related to the number of particles in each cell. Next to this, as already discussed in subsection 3.1.1 and subsection 3.1.2, the robot will also determine the DOA and distance with respect to the robot that sends the message upon receiving the data. The DOA and distance are then used to construct what in this project will be called a **localization table**, of which an example can be seen in Figure 5.1. This localization table is a 2D array containing the possibilities of the robot hearing the other robot while being in a certain cell. Here, the rows represent the possible cell of the listening robot, and the columns represent the possible cell of the sending robot. The values in this table represent the fact that when using a certain combination of cells, the listener robot can hear the sender robot, based on the geometrical distance and angle. In other words, the values of the table represent the possibility that when the listening robot is in cell x, it can hear the sending robot in cell y. An overview of the generation of such a localization table can be seen in Figure 5.2, where a flowchart containing the steps taken in generating such a table are depicted. The first step is determining the true angle (θ_{true}) at which the message is received, by using the current orientation of the robot which is determined by the IMU that is available on the robotic car board. This true angle is calculated using Equation 5.1, where $\theta_{measured}$ is the DOA as determined from the message and θ_{robot} is the robot's angle with respect to the north as determined by the IMU. This true angle is needed in the next steps of the algorithm, to accurately determine the location of the

robot on the map, which takes the true north as a reference point.

$$\theta_{true} = (\theta_{measured} + \theta_{robot}) \mod 360 \tag{5.1}$$

Using the true angle, the program will start looping over all cells twice, processing every combination of cells. For each combination, the shortest and longest distance between the two cells is retrieved, which both have been calculated at the start of the program as described in subsection 4.2.6. Using this information, the first thing that is checked is whether the maximally traveled distance (measured distance + 20) is greater than or equal to the shortest distance and the longest distance is smaller than or equal to the minimally traveled distance (measured distance - 20). If this is the case, it means that the robot could have heard the sender when being in the begin cell, making the cell combination possible when looking solely at the distance traveled by the message. The next thing that is done, is retrieving the path traveled between the two cells and picking the cell that is furthest away and still has a line-of-sight connection with the begin cell. Next up, the angle difference is calculated between the true angle (θ_{true}) and the angle between the begin cell and the chosen cell from the path to the end cell. This is calculated using Equation 5.2, where $\theta_{difference}$ is the angle difference and θ_{cells} is the angle between the begin cell and the chosen cell in the path. Using this angle difference, it is then checked whether this difference is between the margins determined in section 6.2, which is between -25°and 25°. If this is the case, the cell combination is marked as possible in the table (True) and if not, the cell combination is marked as impossible in the table (False).

$$\theta_{difference} = ((\theta_{true} + \theta_{cells} + 180 + 360) \mod 360) - 180 \tag{5.2}$$

Possible cell sender (Robot 1)

Possible cell receiver (Robot 0)	R1	0	1	2	3	...	n
	0	True	False	True	True	-	False
	1	True	False	False	True	-	True
	2	False	True	True	False	-	True
	3	False	False	False	False	-	False
	. . .	-	-	-	-	-	-
	n	True	False	True	False	-	True

Localization table generated inside robot 0 about robot 1

Figure 5.1: Localization table example.

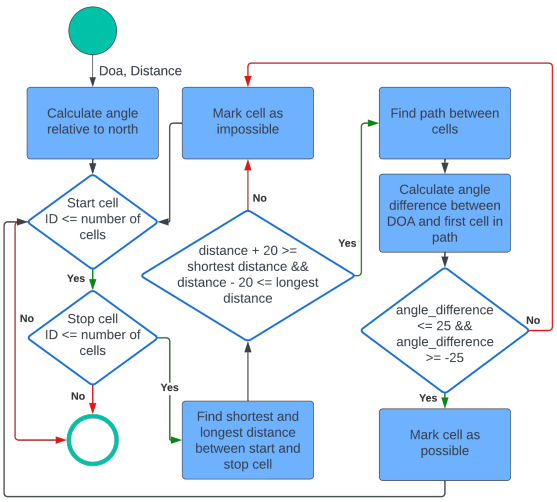


Figure 5.2: Localization table generation flowchart.

Using the generated localization table, the received probabilities per cell from the sender, and the probabilities per cell from the robot itself, the first thing that is done, as can also be seen in Figure 5.3, is combining the data from both probabilities per cell lists using the *outer product*, which results in a 2D array with the same dimensions as the localization table. This array is then combined with the localization table using the *Hadamard product*, which transforms the possibilities into probabilities by multiplying the probabilities calculated in the previous step with either 1 (True) or 0 (False) at the same positions in the table. This new localization table now gives the probability that the listening robot is in cell y and can still hear the sending robot when it is in cell x. This data is then used to calculate the average probability per row, which is an indication of the probability that the listening robot might be in that cell. Lastly, this list of probabilities is normalized, making it suitable for updating the probabilities per cell.

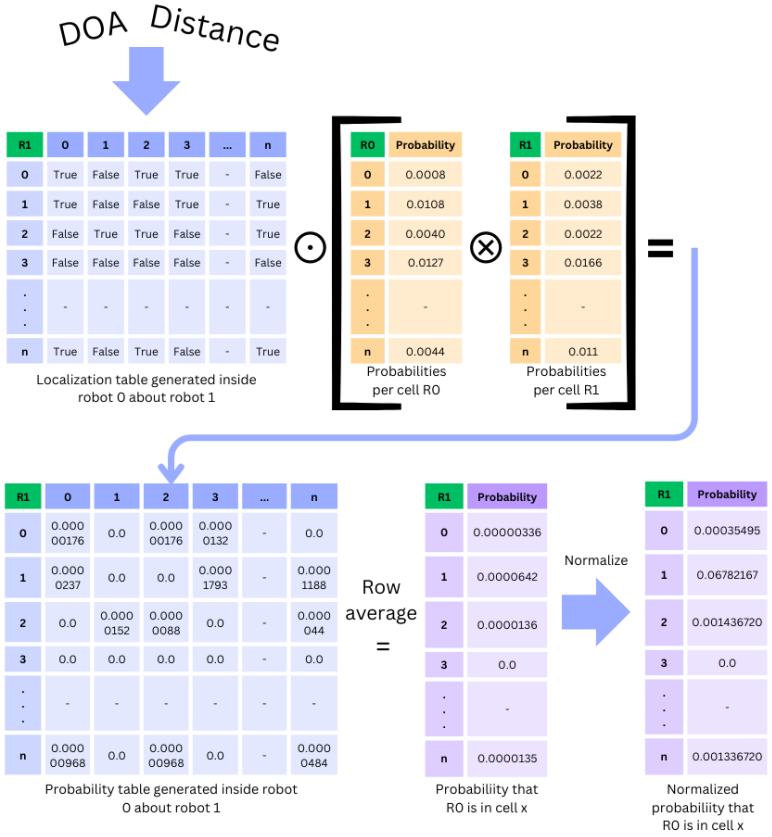


Figure 5.3: Generating localization table on Robot 0 after receiving a message from Robot 1.

Using the calculated average normalized probabilities per row, the next step, as can be seen in Figure 5.4, is using this information to update the current probabilities per cell. This probability per cell data is directly related to the number of particles that are in a certain cell and thus also to the robot's possible location on the map. The new probability is calculated using Equation 5.3, where the probability of the cell is increased by the calculated probability of the row representing that cell. Then, the list of probabilities is normalized to allow for equal particle distribution in the next step. Here, $P(cell_{robot} = X)$ is the probability that the robot is in cell x, and p_{row} is the normalized probability per row as calculated in the previous step.

$$P(cell_{robot} = X) = P(cell_{robot} = X)) + p_{row} \tag{5.3}$$

After determining the new probabilities for each of the cells, some of the particles will need to get resampled from the cells where the probability decreased to the cells where the probability increased. Here, the first thing that is done is determining how many particles need to be in each cell, based on the newly calculated cell probabilities. This is done using Equation 5.4, where $particles_x$ is the new number of particles for cell x, $P(cell_{robot} = X)$ is the probability that the robot is in cell x, and N is the total number of particles. To account for any rounding errors, which could result in more or fewer particles being needed in the cells than the total amount of particles, the amount of particles is either reduced in cells where the number of particles before rounding lies furthest from the rounded number of particles or the number of particles is increased in cells where the number of particles before rounding lies closest to the rounded number of particles in the situation where there are too much or too few particles respectively. Next to this, it is also checked if the row sum, representing the cell in the localization table, is equal to zero, meaning that the listening robot couldn't possibly be in that cell and still be able to hear the sending robot based on the calculated distance and DOA. For these cells, the weight of all particles will be reset to the initial weight, which is calculated using Equation 3.9.

$$particles_x = P(cell_{robot} = X) * N \tag{5.4}$$

Next up, the cells that need to be decreased and the cells that need to be increased are determined by comparing the old number of particles in the cell, with the newly calculated number of particles from Equation 5.4. Then, the cells that need more particles will start to be processed. Here, firstly, the weight of all the original particles in that cell is increased by the initial amount. Next up, the number of particles that need to be added to the cell is calculated and a list of possible particles is generated using the list of cells that need to be decreased. Here, an x amount (the number of particles that need to be removed from that cell) of particles is taken from such a cell after first sorting the particles in that cell on weight in ascending order. Then, for each particle that is moved out of the cell, the weight of the particle is reset to the initial weight. It was chosen to reset the weight of these particles because moving a particle into another cell is so drastic that the weight of the particle cannot possibly represent its age and importance anymore.

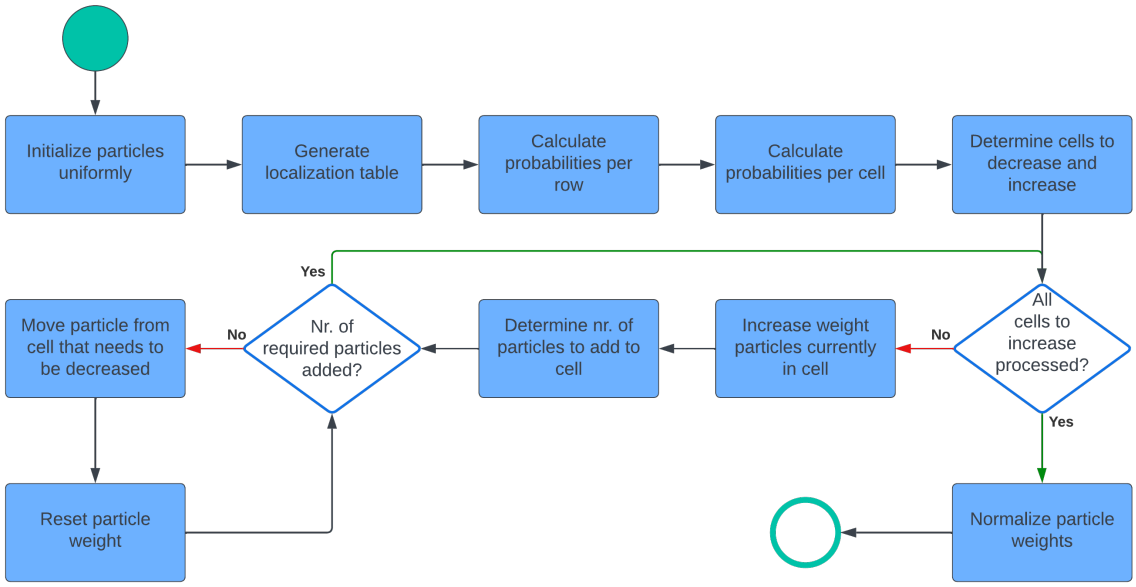


Figure 5.4: Locally generated localization table processing.

2) Receiving a table from another robot about self: The second configuration will help to strengthen the probabilities even more by combining the data received from another robot, removing more uncertainties and measurement errors. In this configuration, it is assumed that the robot receiving the table already has a localization table of the sending robot in its memory, which can be assumed thanks to the order in which the algorithm is executed as can be seen below in Figure 5.7. The process of this configuration can be seen below in Figure 5.5, where robot R0 receives a localization table from robot R1 about himself. The first step that is taken upon receiving the table, is *flipping the table.* Here, the rows and columns are turned, and the sender (R1) and receiver (R0) cells are flipped, resulting in a localization table where the rows represent the sender robot's (R1) cell and the columns the receiver robot's (R0) cells. This flipped localization table is then *overlayed* on top of the generated table that was already in memory about the sender robot (R1), as can be seen in Figure 5.5 where the orange table is the localization table generated by robot R0 about robot R1, and the blue table is the localization table that robot R0 received from robot R1. Here, all cell combinations that are valid in both localization tables will be marked as valid in the resulting localization table, making the table more robust as it is based on the data perceived by both robots. After these steps, the algorithm will continue to function the same as in the first configuration, starting with the calculation of the probabilities per row of the resulting localization table.

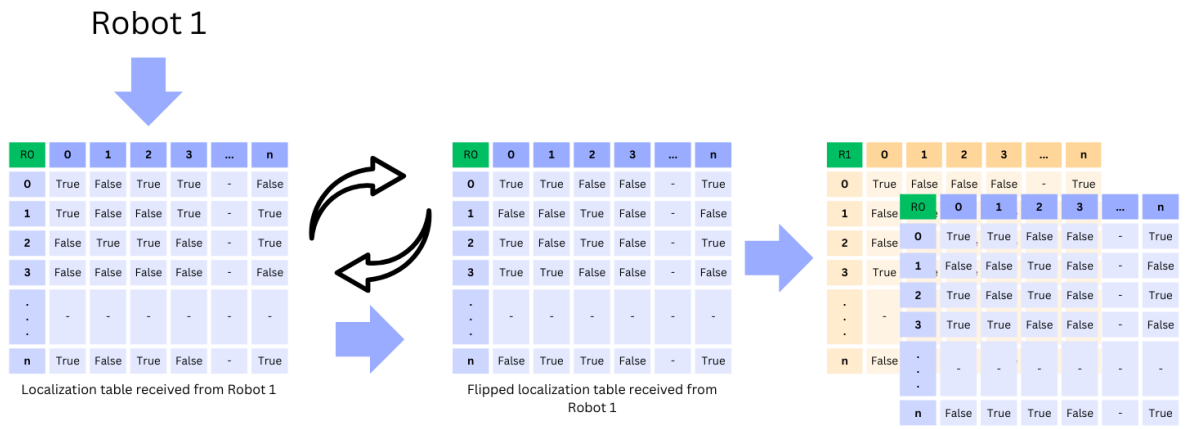


Figure 5.5: Receiving localization table from other about self.

3) Receiving a table from another robot about other: The third and final configuration, in which the robot receives a localization table from another robot that is about another robot, will help to strengthen the information acquired while generating the table for the robot about who the table is received, and the information acquired while hearing from the robot that has send the table. This configuration is depicted below in Figure 5.6, where robot R0 receives a table about robot R2 from robot R1. The first step that is taken, upon receiving the localization table, is summing up all rows and columns. Here, rows that have a sum of zero represent the fact that the sender robot (R1) cannot be in the cell that is associated with that row. This information is thus used to update the table already stored in memory about the sender robot (R1), by marking all fields in the column with the same ID as the row as invalid. The columns that have a sum of zero represent the fact that the robot for who the table was generated (R2) cannot be in the cell associated with this column. This information is then used to update the localization table, if available, of the robot for who the table was generated (R2). If this table is not available beforehand, the information is not used. After updating the localization tables, the tables are again processed in the same way as described in the first configuration, starting with the calculation of the probabilities per row of the resulting localization table.

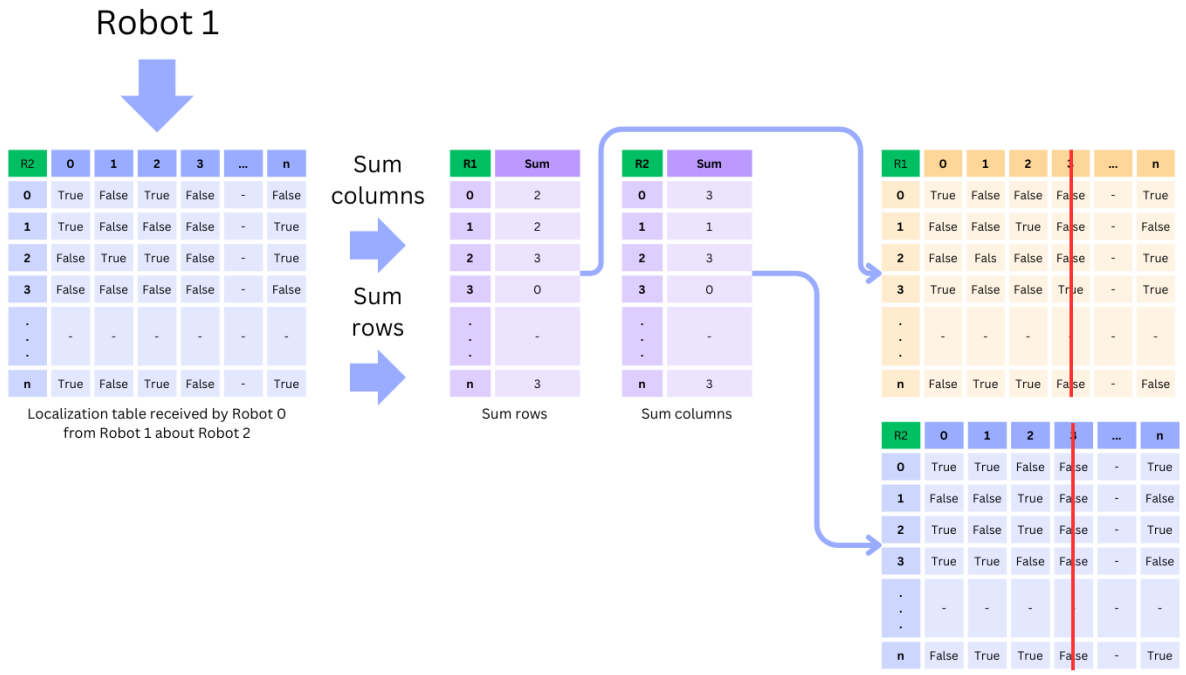


Figure 5.6: Receiving localization table from other about another robot.

During the localization of the robot, a fixed cycle is maintained until convergence has been reached. Here, convergence is achieved when either more than 55% of the particles are in a single cell or the particles are centered around a certain cell with a deviation below 400cm for all particles. These values were chosen after running several tests from which it could be concluded that these values provided the optimal balance between fast and accurate convergence. The fixed cycle that is maintained until convergence can be seen below in Figure 5.7, where a flowchart of the complete cycle is given. Here, the first step of the process is requesting a response from the other robots, waiting for the other robots to respond, and creating and processing a localization table based on this response. This step is finished when the last robot has responded. The robot's response is expected at a certain time based on the known delays for each robot, as described in section 6.3.

All the following steps in the cycle are optional and only executed if the robot is configured to do so. During the evaluation, described in section 6.6, all combinations of these three optional steps have been evaluated. The next step in the cycle is receiving the localization tables from other robots about the robot himself. Here, the robot processes all the received data and waits for the last robot to share its information. Then, during step 3, the tables about other robots are shared and processed in the exact same way. After processing all localization tables, the robot will, just like with a normal particle filter, also be able to drive through the room if the configuration allows it. During driving, all particles will be moved in a certain direction and distance, improving the localization accuracy even more. Here, the robot will move exactly one cell, which is chosen based on the particle distribution. Here, the cell with the highest probability is chosen, and based on the path to that cell, the robot drives to the first cell in that path. During the last step of the cycle, it is checked if convergence has been reached and if that is the case, the cycle will be terminated, and the robot's position will be highlighted on the map. If this is not the case, however, the cycle will be repeated from the start until convergence has been reached. Thanks to this fixed order of execution, the tables that are received from other robots during the second and third scenarios can be combined with the tables already generated by the robot. This is a valid assumption since the robot has not moved and the fact that a table has been received from the robot means that the robot is within range, which means that it should have been heard during the first step of the cycle.

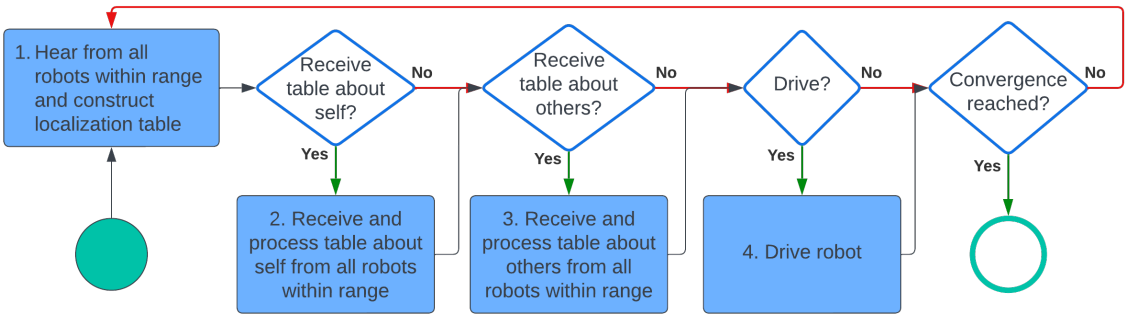


Figure 5.7: Localization cycle used by the algorithm.

6

Evaluation

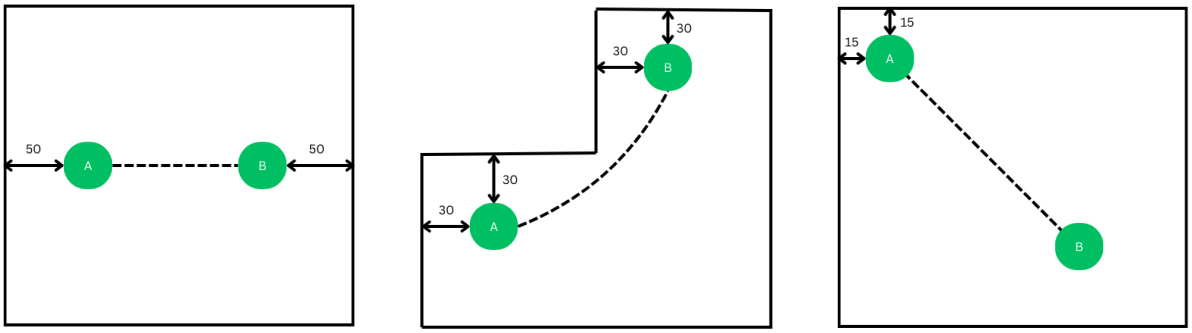
This chapter will contain the evaluation work of this thesis. First, the existing techniques will be evaluated, namely the direction of arrival and distance. Next up, the message encoding and decoding schema will be evaluated using varying number of listeners and distances. This is followed by the evaluation of the particle filter. Lastly, the newly proposed algorithm of this thesis will be evaluated.

6.1. Evaluation setup

Three evaluation configurations were defined to evaluate all audio-based parts in this thesis (DOA, distance, message encoding). These three configurations should cover most situations in which the robot could be in a realistic indoor scenario. These configurations are explained here and will only be mentioned by their abbreviation throughout the rest of the chapter. First up, there is the **Line-of-Sight (LOS)** (Figure 6.1a) configuration, which is also known as the *free space* configuration. In this configuration, there are no obstructions between the robots, and the robots are placed at least 50cm from any obstructions in the room to prevent any reverberation. This configuration will most likely not represent a realistic scenario for any indoor environment, but it is a good way to show the performance under ideal circumstances. Next up, there is the **Non-Line-of-Sight (NLOS)** (Figure 6.1b) configuration, where the robot's direct line-of-sight is obstructed by, in this case, a wall. Here, the robots are placed around the corner from each other, at least 30cm from any wall, meaning that the sound signal must travel around that corner to reach the other robot. This configuration represents a very realistic scenario, as robots in different rooms will be in a similar situation. Last up, there is the **Reverberation (Reverb)** (Figure 6.1c) configuration, where one of the robots is placed into a corner with only 15cm distance to each of the two walls, while the other robot is placed in a LOS position. Because the robot is now so close to the wall, it will suffer from reverberation effects where the signal is reflected from the wall, creating a delayed echo of the original signal [117]. This results in the forming of multiple paths of the same signal, which could cause trouble for both sending and receiving messages. Within these three configurations, measurements were taken starting from 10cm for the LOS and Reverb configurations and 25cm for the NLOS configuration, up to 350cm. The NLOS configuration measurements start from further away because it wasn't possible to construct an NLOS scenario at 10cm. These three evaluation configurations are summarized below in Table 6.1.

	LOS	NLOS	Reverb
Line-of-Sight	x		x
Distance to wall robot A	> 50cm	>30cm	15cm
Distance to wall robot B	> 50cm	>30cm	>30cm
Minimum evaluation range	10cm	25cm	10cm
Maximum evaluation range	350cm	350cm	350cm

Table 6.1: Evaluation configurations.



(a) Line-of-Sight (LOS) **(b)** Non-Line-of-Sight (NLOS) **(c)** Reverberant (Reverb)

Figure 6.1: Evaluation positions.

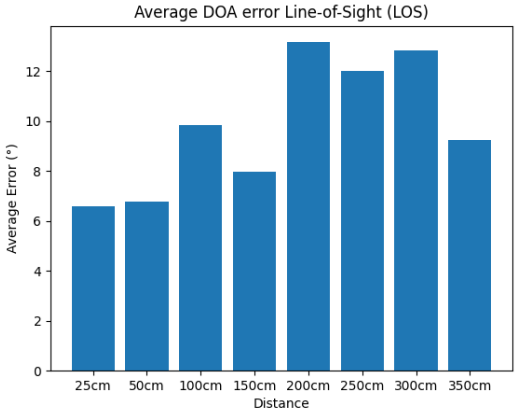
6.2. Direction of Arrival

The direction of arrival was evaluated using two robots, where the algorithm described in subsection 3.1.1 was used to calculate the DOA based on the arrival times of the same message at each of the six microphones. An extensive evaluation has been carried out to get a comprehensive view of the DOA algorithm's performance. During the evaluation, two robots were used and placed in either one of the three configurations described above in section 6.1. For each of these configurations measurements were made at various distances, starting from 25cm up to 350cm with intervals of 50cm. For each distance, 100 measurements were executed at four angles, namely 0°, 90°, 180°, and 270°. This resulted in a total of *400 measurements* per distance and a total of *3600 measurements* per configuration. The choice for this evaluation setup, where four different angles are evaluated, was made so that the results reflect a more realistic scenario where the message can be received from different directions. Since the microphone array is circular, performing measurements at each distance for four different angles will ensure that different microphones face closest to the other robot each time. The choice for these specific four angles was made because it makes sure every time another face of the robot faces the source robot and because it allows for easy alignment. Since the robot's body is rectangular, taking angles of 90°apart allowed for easy alignment without the need for advanced measurement tools.

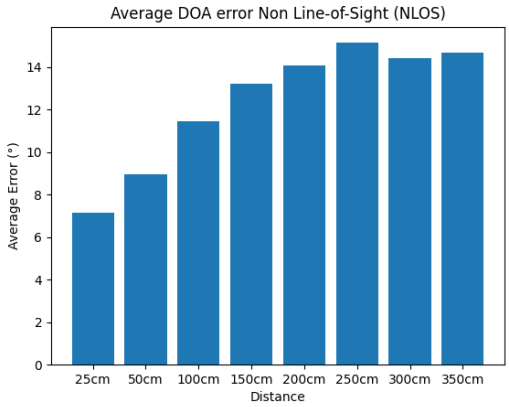
The results of these measurements, which are given in the form of the average DOA angle error in degrees, can be seen in Figure 6.2. Here, the average angle error was chosen as this reflects the overall accuracy of the algorithm properly, and it allows for easy identification of the error margins that will be used in the proposed algorithm. First, the LOS configuration results can be seen in Figure 6.2a, where a maximum average angle error of 13°can be observed. Another thing that can be observed is that the results seem to fluctuate over distance, where lower distances sometimes have a higher average error than higher distances. This contrasts with the NLOS and Reverb configurations, where a linear relation can be found with respect to the distance. This configuration also has the best results out of the three configurations, which can be explained by the fact the LOS signal in this configuration will almost always dominate the signal as there are hardly any reflections. Although the linear relation to the distance is difficult to see, it can be observed that, on average, the error increases as the distance increases.

The results of the NLOS configuration can be seen in Figure 6.2b, where the average error is greater than or equal to the errors observed in the LOS configuration for every distance. This is in line with the expectation since more reflections can occur for the NLOS configurations because the signal needs to bend around a corner, off which it will also be reflected, to reach the receiver. This effect impacts the DOA performance even more as the distance increases, as the reflections have an even greater chance of reaching the receiver first in this case. The maximum average error for this configuration was found to be *14.5°*. Last up, there is the Reverberation configuration, which can be seen in Figure 6.2c, where a maximum average error of *22.5°* is observed. Here, the relation between the distance and the average angle error follows the exact opposite trend than for the LOS and NLOS configurations, where the average error decreases with the distance instead of increasing. This can be explained by the fact that when the two robots are close together, it also means that they are both close to the reverberating

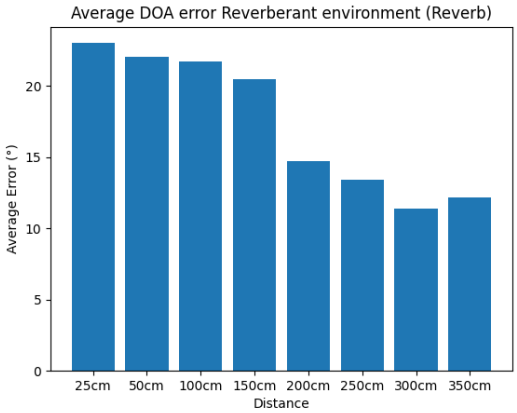
source, allowing reflections to reach the receiver robot more easily. As the distance increases, the chance of the reflections reaching the robot faster than the direct LOS signal becomes smaller and therefore the error decreases. The fact that the average error of the higher distances is better than that of the NLOS configuration can be explained by the fact that the robots are in an LOS arrangement.



(a) DOA error LOS.



(b) DOA error NLOS.



(c) DOA error Reverb.

Figure 6.2: DOA evaluation results.

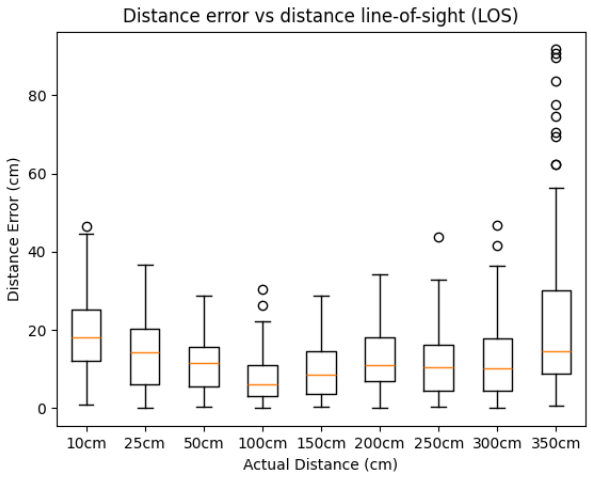
Based on these measurements, it can be concluded that the chosen algorithm is imperfect, and large errors of up to 22°can occur. Other algorithms, like machine learning, could improve the performance of the DOA estimation, which would come at the cost of the processing power, of which there is a limited amount on a mobile platform like the robotic car. Fortunately, the proposed algorithm is designed to work with inaccurate results, allowing this computationally friendly DOA algorithm to be used for this project. From the results, a margin of **25°** was chosen for the algorithm, which goes both ways. This results in a field of *50°* around the measured angle, in which it is assumed that the true angle should lie.

6.3. Distance

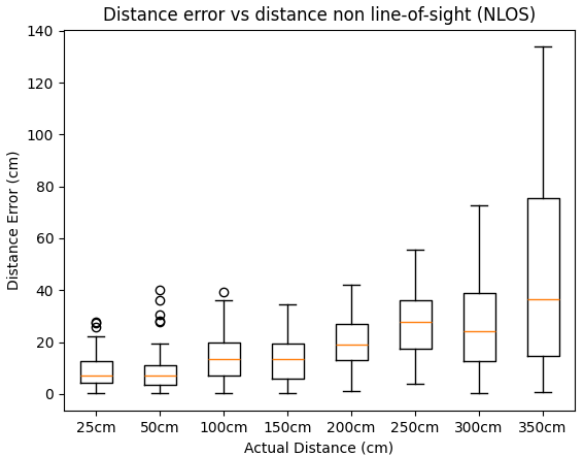
The evaluation of the distance measurement was, just like that for the DOA, done using two robots and the algorithm described in subsection 3.1.2. These robots were placed in the exact same situations as described in section 6.1. For both the LOS and Reverb configurations, robot A was placed in a stationary position while robot B was moved with intervals of 50cm. For the NLOS configurations, both robots were initially placed around a corner at 25cm apart. The robots were then moved apart equally, increasing the range by 50cm. At every distance and configuration combination, a total of *100 measurements* were made, resulting in *2600 measurements* in total. This number of measurements

was chosen to find a real representation of the average error, as the results seem to fluctuate with each measurement, and this would smooth out heavy outliers.

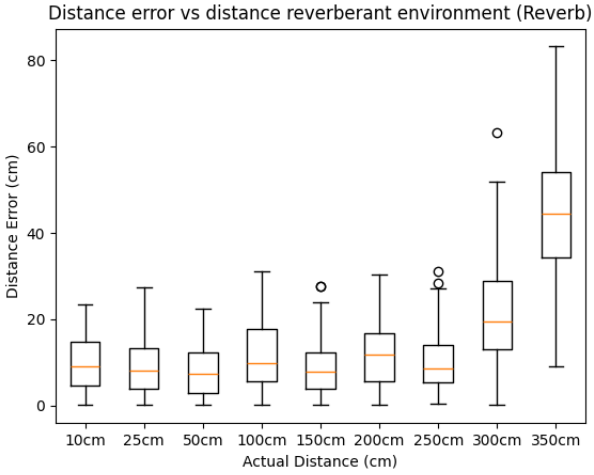
First up, the results of the LOS measurements can be seen in the box plot in Figure 6.3a. Here, the average error is smaller than 20cm and the further the distance increases, the more susceptible the algorithm becomes to outliers. At distances of 350cm or higher, the error rate becomes significantly high, reaching over 100cm, making the algorithm unusable at these distances. This effect can be explained by the fact that the signal energy degrades over distance, and because the hardware can only output a certain amount of signal energy to begin with, there will be almost no signal energy left at large distances, making it impossible to detect a message. The NLOS measurement results can be seen in Figure 6.3b, which for the smaller distances is only slightly worse than the LOS configuration. For the larger distances, however, the NLOS configuration seems to perform significantly worse, which was to be expected since it needs to travel around a corner which it also reflects off. This lowers the signal energy even more than only the distance, resulting in worse performance. Lastly, the Reverb measurement results can be seen in Figure 6.3c, where at the smaller distances the results are about the same as that in the LOS configuration. At larger distances, the performance again seems to degrade and fluctuate more. This could, however, also be explained by the fact that robot B was coming increasingly closer to a wall at this point, introducing even more reverberation into the equation.



(a) Distance error (cm) LOS.



(b) Distance error (cm) NLOS.



(c) Distance error (cm) Reverb.

Figure 6.3: Distance evaluation results.

To conclude, the average error in all three configurations lies below 20cm, and only at higher distances it increases significantly. Based on this evaluation, a margin of **20cm** was chosen in the proposed algorithm, resulting in a radius of *40cm* around the measured distance. This should account for most of the faulty errors and ensure the proper functionality of the proposed algorithm. As mentioned before during the DOA evaluation, these faulty measurements are not a big problem as their inaccuracy has been considered in the proposed algorithm, allowing it to work even when the SSL is not working optimally.

6.4. Messaging codec

To provide a complete evaluation of the performance of the messaging codec, the three configurations, as described at the start of this chapter, will be used during the evaluation. However, before the evaluation of the messaging codec can be performed, the optimal number of samples to be used for encoding the preamble and bits needed to be determined. Here, using more samples to encode a single bit will make it more resilient against noise. This, however, does come at the expense of the data rate, which will decline when using more samples. Because of this, firstly the number of samples that will be used to encode a single bit was determined. This was done by determining the average Bit Error Rate (BER), which is calculated using Equation 6.1, over *100 runs*, while using a different number of samples to encode the bits and using different values of signal-to-noise ratio (SNR). The results of this evaluation can be seen below in Figure 6.4 and Figure 6.5, where five different number of samples are tested against nine different SNR values. Here, each of the five different numbers of samples needed to be dividable by twelve, as each bit consists out of twelve sub-chirps as described in subsection 4.2.4. Looking at these results, the performance of the encoding schema is quite good, where the BER is smaller than 2% up to -10dB SNR for all number of samples. Based on the results, it was determined to use **768 samples** to encode a single bit. With this number of samples, a still reasonable data rate of **57.42bps** can be achieved while still working under very noisy conditions. Next to this, this number of samples was tested to be the lowest amount for which distances of up to 3 meters could be achieved, which is desirable for the proposed algorithm of this thesis.

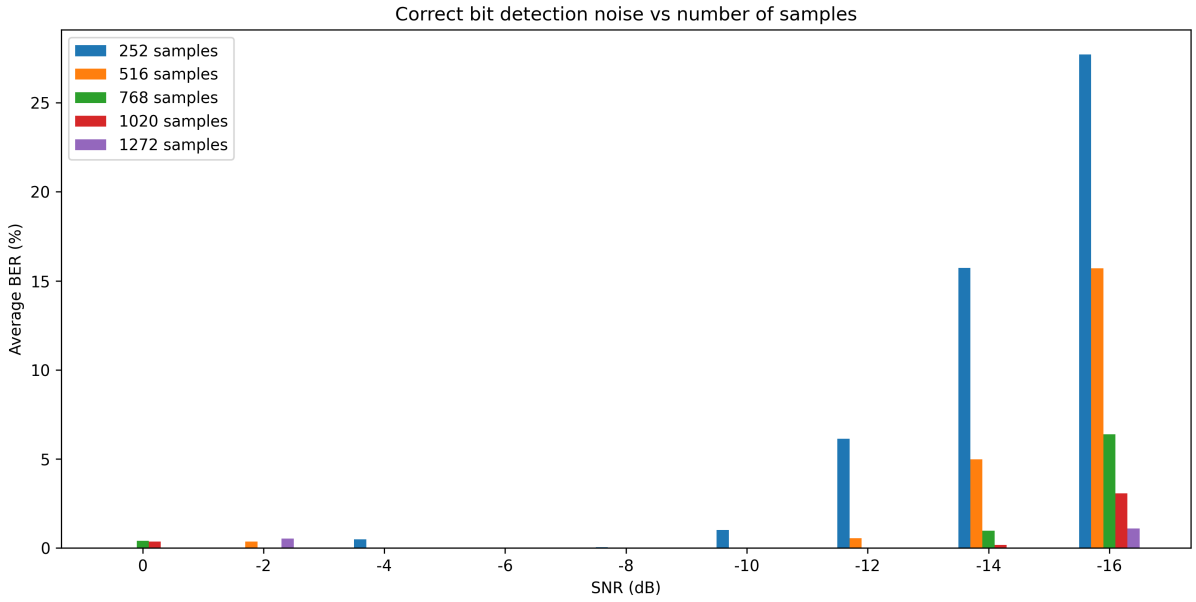


Figure 6.4: SNR vs average BER.

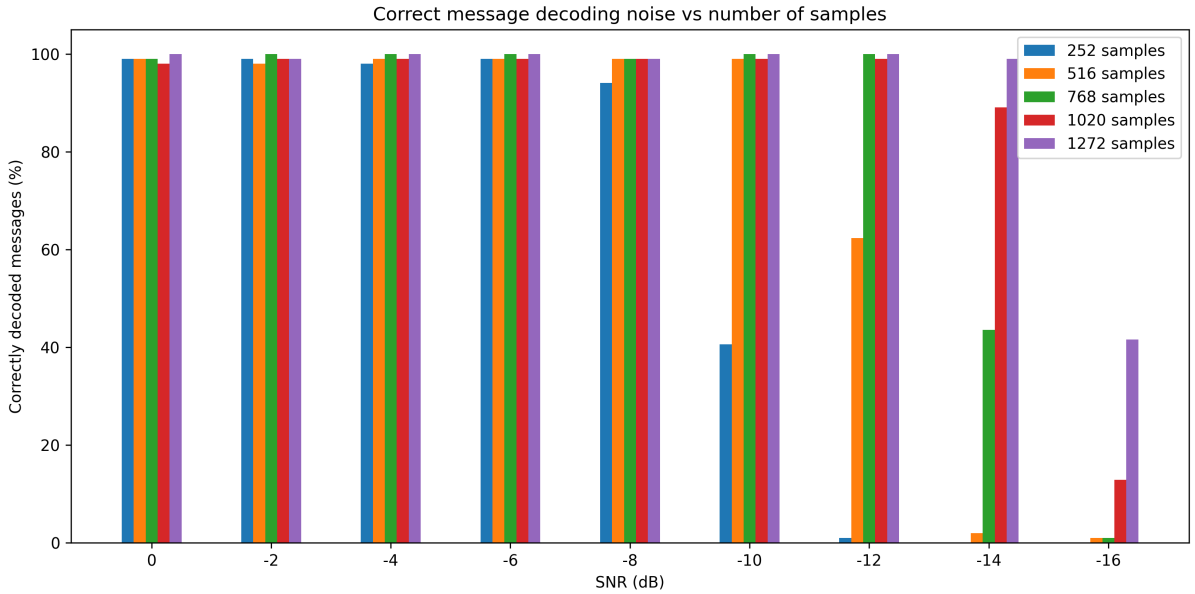


Figure 6.5: SNR vs correct message decoding.

After determining the number of samples to be used for encoding a single bit, the number of samples for encoding the preamble needed to be determined. The preamble is an important part of the messages, as it signals the start of the message, synchronizes the message, and allows for the determination of the DOA. Because of its importance, it is key that the preamble can be correctly detected for every message. To determine the optimal number of samples for this to be achieved, eight different amounts of samples are compared to fourteen different SNR values, where for each combination, the percentage of correctly detected preambles is measured over *100 runs*. The results of this can be seen below in Figure 6.6. Looking at these results, it becomes clear that the preamble can still be detected successfully under very noisy conditions, where using 1024 samples still results in a detection rate of 100% up to -8dB SNR.

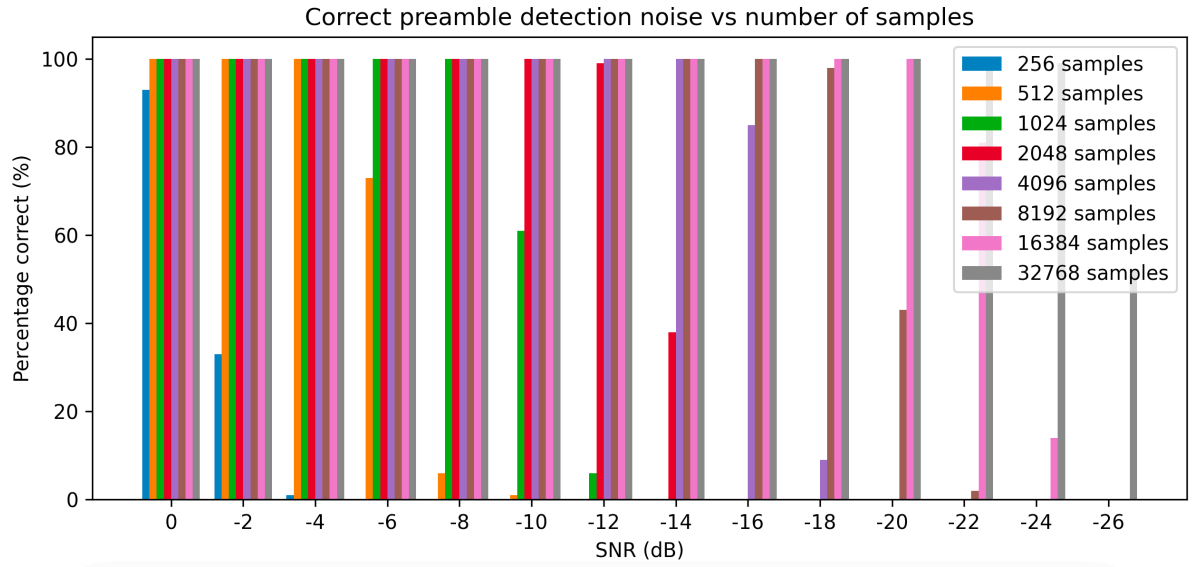


Figure 6.6: SNR vs correct detection of the preamble.

Another important factor to consider when choosing the optimal number of samples to encode the preamble, is the computational complexity of increasing the number of samples. As can be read in subsection 4.2.5, a matched filter is used to decode the received messages, which has a complexity of O(2n log n). Because of this high complexity, the effect of under-sampling was also researched, as this would allow the use of a large preamble size while decoding it using a smaller size. Here an under-sampling divisor was chosen between one and four for the sample values between 256 and 8192 samples. The results of this test can be seen below in Table 6.2, where the lowest SNR value is given for which the preamble is still detected at least 95% of the time. Here, under-sampling does reduce the noise resilience of the message, but with the added benefit that the size of the preamble in the message is still the original size making it less prone to interference during transmission. Based on these results, it was chosen that **8192 samples** will be used to encode the preamble and that an under-sampling divisor of **four**, resulting in **2048 samples**, will be used during the detection of the preamble, allowing for four times faster detection results than when using the original size. Next to this, just like with the sample size for the bit encoding, this sample size works almost flawlessly until **-12dB SNR**, ensuring that both parts of the message cannot be a bottleneck of one another. Finally, using *2048 samples* was also found out to be the upper limit when decoding messages in real-time, as any higher resulted in buffer overflows and loss of data.

	Original samples		
Under sampled samples	2048	4096	8192
512	-4dB	> 0dB	-
1024	-8dB	-6dB	-6dB
2048	-12dB	-12dB	-12dB
4096	-	-14dB	-14dB
8192	-	-	-18dB

Table 6.2: Under-sampling results.

After determining the optimal number of samples for both the preamble and bits encoding and the implementation of the codec, the next thing that was done was evaluating the actual performance of the messaging codec. As can be read in subsection 4.2.5, each robot uses a unique pattern to encode its bits. This means that decoding these bits depends on the message's sender, which in turn means that decoding the robot ID part of the message successfully is crucial. When this part of the decoding goes wrong, the bits will be decoded using the wrong bit encoding as a reference, making decoding the message impossible. Because of this, the first part of the messaging codec that is evaluated is decoding the robot ID section of the message. This evaluation is done by decoding a test message *100 times* while placing the robot at various distances between 50cm and 300cm, varying the number of robots in the swarm between two and six, and using one of the three configurations as described in section 6.1. Here, for each run it is checked if the correct robot ID is detected and based on this information the probability is calculated for each combination of distance and number of robots, where the probability is averaged over the three configurations.

The results can be seen below in Figure 6.7, where the probability of detecting the right robot ID is shown. Looking at the results, it becomes clear that the biggest impact on the performance is the number of robots deployed in the swarm, whereas when there are only two or three robots deployed in the swarm, a 100% success rate can be achieved. Next to this, the distance is also an import factor, where at 300cm the performance drops significantly when deploying five or six robots in the swarm. Another thing that can be noticed is the slightly lower decoding performance at the lowest distance, which can be explained by the reflections caused in the reverberating environment. For both the LOS and NLOS configurations, the results of the 50cm distance were consistently better than those at 100cm. To conclude, even in the worst case of having six robots at 300cm, the messaging codec can still decode the correct robot ID with 90% probability, showing it has a great performance. Next to this, when detecting the wrong robot ID, it is always just one off, meaning it finds the neighboring robot ID. This fact could be used in future work, when a large BER is detected, to try to decode the bits again using a neighboring robot's decoding schema.

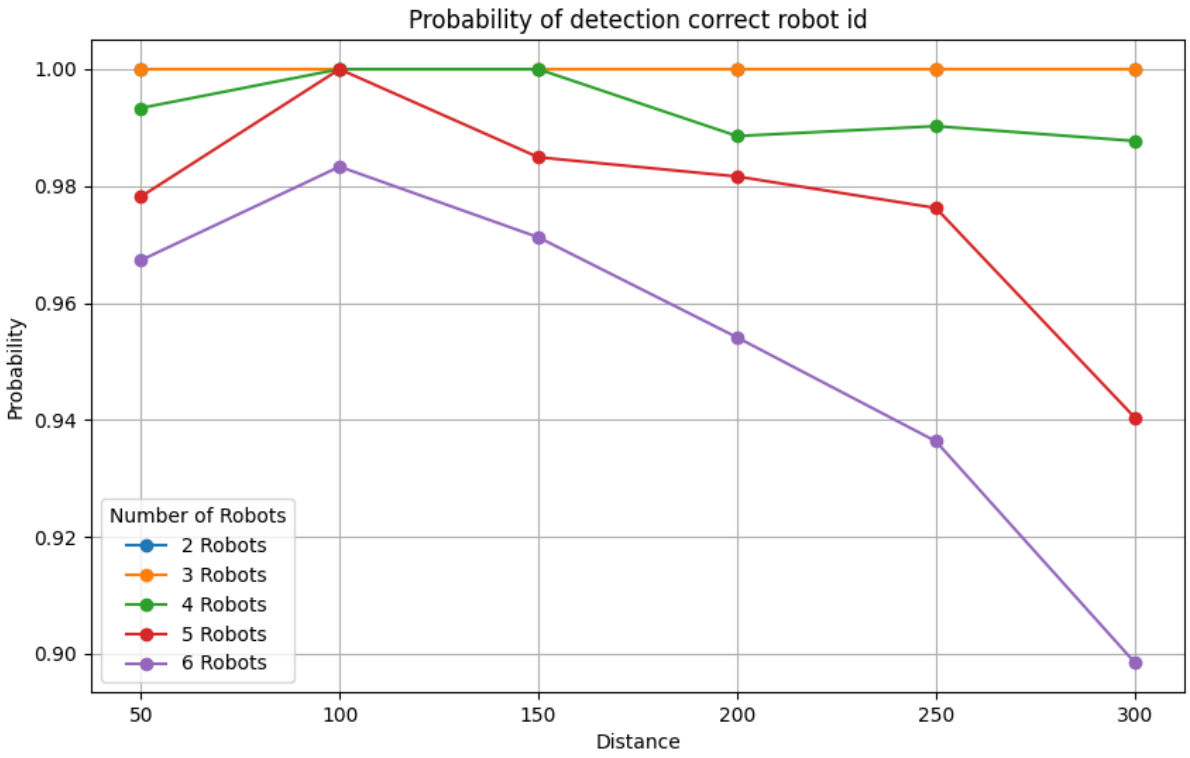
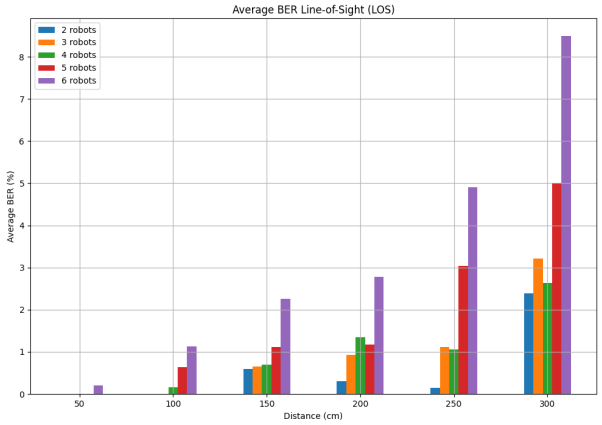


Figure 6.7: Probability of decoding the correct robot ID.

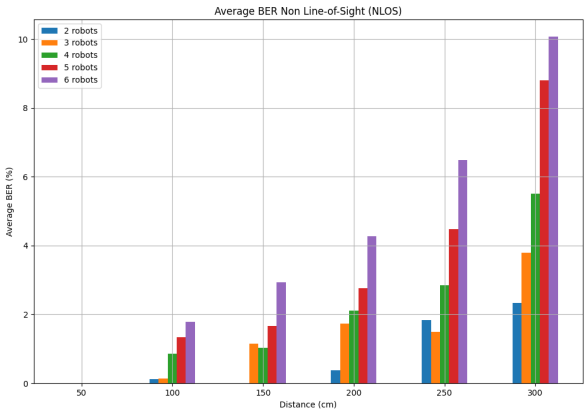
After finding out that the robot ID can be successfully decoded with high probability, the decoding of the rest of the message was evaluated. This is done by determining the BER when receiving a test message from various distances between 50cm and 300cm, varying the number of robots in the swarm between two and six, and using one of the three configurations as described in section 6.1. Here, the BER is calculated using Equation 6.1, where N_e is the number of wrong bits and N_t is the number of transmitted bits. The number of wrong bits (N_e) is determined by receiving a test message from another robot and comparing the received bits with the original sent bits.

$$BER = \frac{N_e}{N_t} \tag{6.1}$$

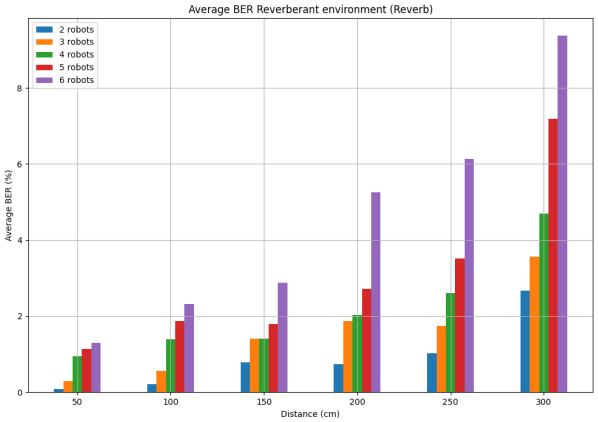
In the same way as during the evaluation of the robot ID detection, a message is being decoded *100 times* for each unique distance, number of robots, and configuration combination. Here, upon receiving each message, the BER is calculated and in the end the average BER is calculated over all 100 runs for each unique combination. First, there are the LOS results, which can be seen in Figure 6.8a. Here, it can be observed that both the number of robots and the distance have a negative impact on the performance. Another thing that can be observed is that the performance at 300cm is significantly worse than at lower distances, a trend that increases even more when increasing the distance even further than 300cm. Because of this, the evaluation was limited to 300cm. Next up, the results of the NLOS configuration can be seen in Figure 6.8b, where the same trend and a slightly bigger error can be observed than for the LOS configuration. This can be explained by the fact that there is no direct line of sight, which means that the message needs to bend around a wall, by which it also partly gets absorbed and reflected off. Lastly, the results of the reverberation configuration can be seen in Figure 6.8c, which show worse performance at the lower distances than the other two configurations. This can again be explained by the fact that the signal power of the reflections is much higher when both robots are close to the reverberation source. Looking at all three configurations, it can be concluded that the messaging codec can perform quite well in terms of average BER, where, on average, a BER of only 2% is achieved, and even in the worst-case scenario, the BER is just 10%.



(a) Average BER (%) LOS.



(b) Average BER (%) NLOS.



(c) Average BER (%) Reverb.

Figure 6.8: Messaging codec BER results.

The last and most important factor evaluated for the messaging codec is the probability of decoding a message successfully. This is an important metric as it tells exactly how useful the messaging codec is, as even a BER of smaller than 1% results in a message not being decoded properly, resulting in the data being lost. To evaluate this metric, the same evaluation setup is used as mentioned above, where the messages with a BER higher than zero are marked invalid. The results of this evaluation can be seen below in Figure 6.9, where the probability of successfully decoding a message is given for a varying number of robots and distances. Here, the average result is taken over the three configurations to give an overview of the overall performance of the messaging codec. Looking at the results, it can be concluded that both the distance and number of robots in the swarm have a big impact on the performance of the codec, where at 300cm almost no messages are decoded properly anymore with the highest probability of only 60%. This graph shows that even when the BER is close to zero, as was concluded from the plots in Figure 6.8, it is no guarantee that messages get decoded successfully, since a one-bit error already means that the message is not successfully decoded.

These small bit errors could, in future works, be prevented by using something like error correction schemes. However, since those increase the number of bits in one transmission in combination with a small data rate, it was chosen to consider this out-of-scope for this thesis project. To conclude, the messages up to 200cm are decoded with a probability of around 80%, which is a reasonable performance. However, when increasing the distance further, especially up to 300cm, the performance of the messaging codec drastically worsens and becomes unusable, where it drops as low as 20% in the worst case, resulting in most messages being dropped. This, however, is not terrible for the work proposed in this thesis, as the distance and DOA information can still be extracted. This means that the message can still be used to update the robot's expected position and therefore the message is not a total loss.

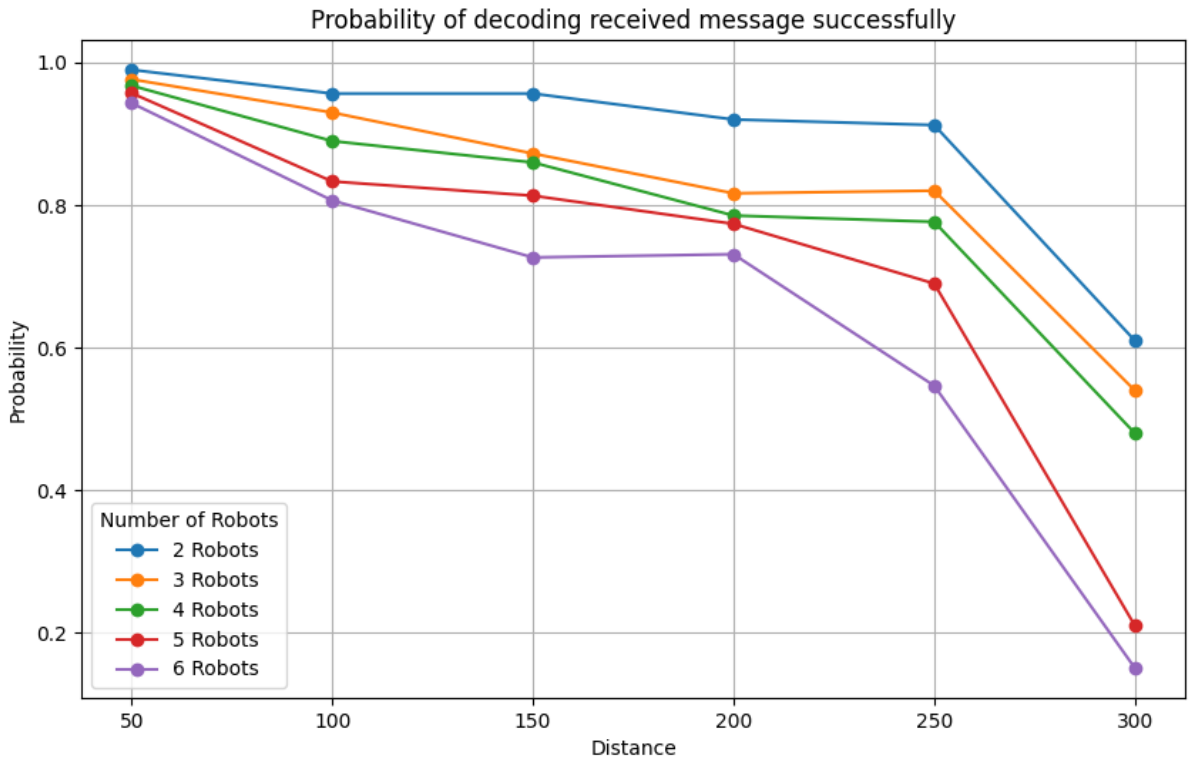


Figure 6.9: Probability of decoding message successfully.

6.5. Particle filter

To get an overview of the performance of the particle filter before it's combined into the proposed algorithm of this thesis, an extensive evaluation was executed. Here, the performance of the particle filter in combination with the motion model was evaluated by driving the robotic car through the room in different patterns. Based on the works in [118–122], where also a particle filter or particle filter fusion algorithm is implemented, the choice was made to evaluate the particle filter based on the Root Mean Square Error (RMSE), convergence speed, the error probability, and by tracking a certain path after achieving convergence. In each iteration, the particles will be moved *39.72cm*, which is exactly one complete wheel rotation for the robotic car, as deduced in section 3.3. This means that for every iteration, every particle will be moved exactly 39.72cm from its previous position. To achieve the most realistic results, four different paths were chosen for the robot to drive and achieve convergence. These paths represent the fact that the robot initially does not know where it is and starts to drive in a random direction as can be seen in Figure 6.10, where each color represents a different path. During the construction of these paths, it became clear that the robot needs to drive at least **628.32cm** before it can achieve convergence, based on the robot's maximum speed, which can be found in Table 4.1, that would take around **3.2 seconds**. Here, the convergence is determined by looking at the deviation of all particles and if that number is below a certain threshold (400cm) for both the x and y coordinates of the particles. Using the same methods, the average distance that needs to be traveled was also determined to be **908.12cm**, which translates to around **4.62 seconds**.

After constructing the paths that the robot will drive to achieve convergence, the first thing that was measured is the *Root Mean Square Error (RMSE)*, which is the mean of the squared error displaying the difference between the predicted and observed locations of the robot. This metric was chosen as it gives a good measurement of the performance of the particle filter in a single number. Next to this, a lot of the other works using particle filters also use this metric, making it easy to compare to these works and the proposed algorithm of this thesis. The RMSE is calculated using Equation 6.2, where I is the number of iterations taken in completing one driving sequence, $x_{real}(k)$ is the real position of the robot, and $x_{estimated}(k)$ is the estimated position of the robot.

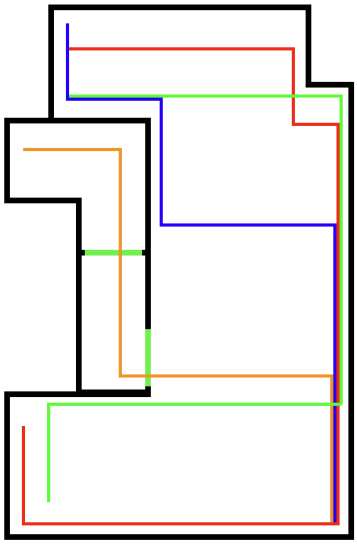


Figure 6.10: Convergence paths used in the evaluation.

$$RMSE = \sqrt{\frac{\sum_{k=0}^{I}(x_{real}(k) - x_{estimated}(k))^2}{I}} \tag{6.2}$$

The estimated position of the robot is based on the location and the normalized weight of the particles. These two are combined to calculate the robot's possible x and y coordinates on the map using the *Weighted Sum*. To achieve this, Equation 6.3 and Equation 6.4 are used to calculate the possible x and y coordinates respectively, where N is the total number of particles, x_k and y_k are the x and y coordinates of the particle respectively, and w_k is the weight of the particle.

$$S_x = \sum_{k=0}^{N} w_k * x_k \tag{6.3}$$

$$S_y = \sum_{k=0}^{N} w_k * y_k \tag{6.4}$$

During the evaluation, the first thing that is done is following one of the four paths completely, after which the robot will drive in a certain figure to test the performance of the particle filter after convergence. Each of these four paths is evaluated *50 times*, resulting in a total of *200 measurements* used for the evaluation. With the help of the equations above, the RMSE of the algorithm was measured to be **256.67cm** before achieving convergence and **15.96cm** after achieving convergence. Based on these results, it can be concluded that the performance of the particle filter is not good before achieving convergence, which is to be expected since the robot initially has no way of knowing where it is. After achieving convergence, however, the performance of the particle filter drastically increases, and high accuracy localization becomes possible. These results can also be seen below in Table 6.3, where the evaluation results for the particle filter are displayed.

	Before convergence	After convergence
RMSE	256.67 cm	15.96 cm
Min distance until convergence	628.32 cm	-
Max distance until convergence	1806.42 cm	-
Average distance until convergence	908.12 cm	-

Table 6.3: Particle filter evaluation results.

Next to the RMSE, also the error distribution for both situations was measured by recording the distance error between the actual location of the robot and the measured location of the robot using the Euclidean distance as calculated by Equation 4.4. This data is plotted below in Figure 6.11 and Figure 6.12, where the *Cumulative Distribution Function (CDF)* is given for both scenarios. From these plots, it can be deduced that more than 50% of the errors are below **60cm** when convergence is not yet achieved and below **16cm** when convergence has been achieved. This indicates a good performance, as the probability that the error is small is high. The CDF was chosen to evaluate the error because it gives good insights into the probability that the distance error is smaller than a certain value and how often it occurs. Next to this, it also handles outliers very well as these will simply be moved to right of the plot and won't affect the results too much resulting in a good performance overview.

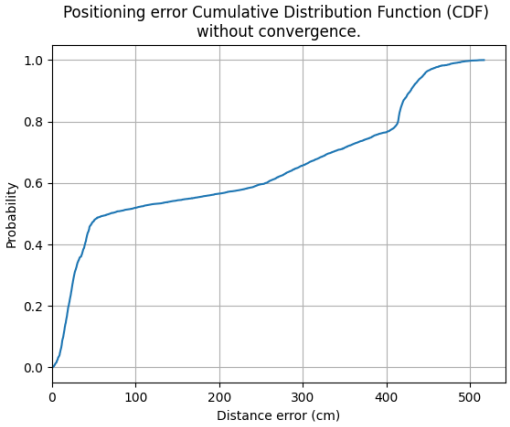


Figure 6.11: CDF localization error particle filter before convergence.

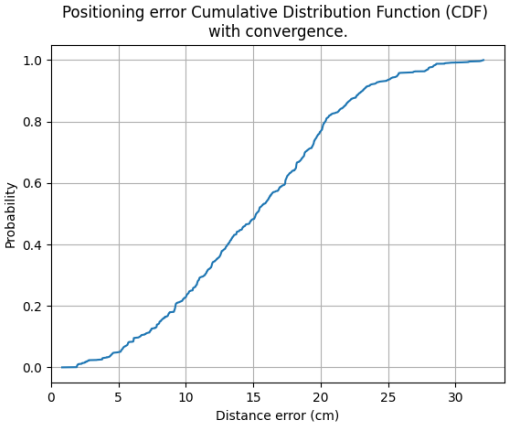


Figure 6.12: CDF localization error particle filter after convergence.

The last part that was evaluated for the particle filter is letting the robot drive a certain pattern after convergence is achieved and comparing that to the measured location of the robot at every step. For this, a rectangular and triangular path was chosen, as seen in Figure 6.13 and Figure 6.14 respectively. Here, both paths have a small tail, which is the route the robot had to drive to the start position of the figure. From these results, it can be concluded that the robot can track the path nicely, where only relatively small errors are observed while driving. Combining all evaluation data, it can be concluded that the particle filter's performance is highly dependent on how well the convergence stage is executed, as poor convergence leads to poor performance after convergence. To conclude, the particle filter implemented in this project has a pretty good performance, if convergence is executed properly, and it can reach convergence in a reasonable amount of time.

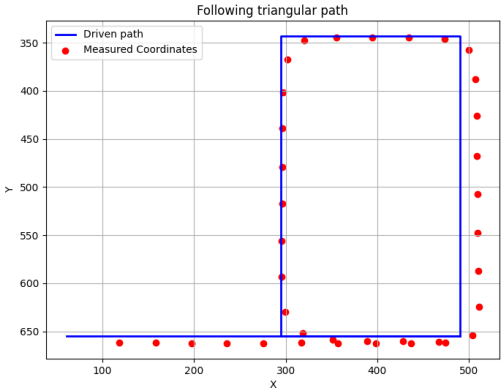


Figure 6.13: Tracking of rectangular path.

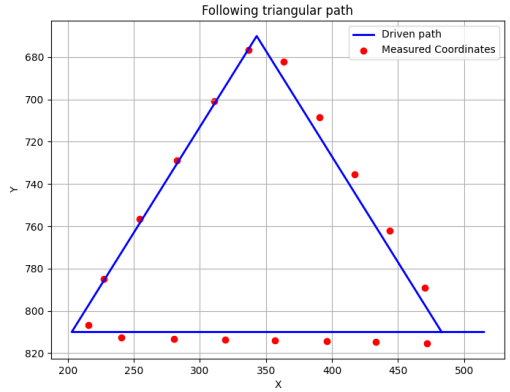


Figure 6.14: Tracking of triangular path.

6.6. Fusion of PF and SSL techniques

As mentioned before, the main contribution of this thesis work is the algorithm that combines the particle filter and sound source localization techniques to achieve high accuracy indoor localization. To discover how accurate and fast this localization can be achieved using this novel method, an extensive evaluation of the algorithm has been executed. The evaluation has been performed for two different scenarios: 1) the robots are *not allowed* to drive during localization, and 2) the robots are *allowed* to drive during localization. For both scenarios, all three different types of configurations as discussed in section 5.1: hearing from another robot (normal), receiving a table from another robot about self, and receiving a table from another robot about another robot, are evaluated separately. Here, each additional configuration is combined with the configuration below it, meaning that when evaluating the second configuration, the first configuration will also be active and data acquired by hearing other robots is still processed. This results in the last configuration embodying the total system, where all configurations are active. The choice to evaluate the algorithm in this way was made because it would create an overview of the possible advantages or disadvantages of using the different configurations. For each of these configurations, the number of robots is also varied *between three and six*, which was chosen because using less than three robots results in really poor performance, where convergence is mostly not reached, and using more than six robots is not possible when using the messaging codec due to hardware limits, as can be read in section 6.4. Each of these 36 unique configurations has been evaluated exactly **100 times** by placing the robots into random cells. By placing the robots into random cells, the possible advantages or disadvantages of the location are averaged out in the results.

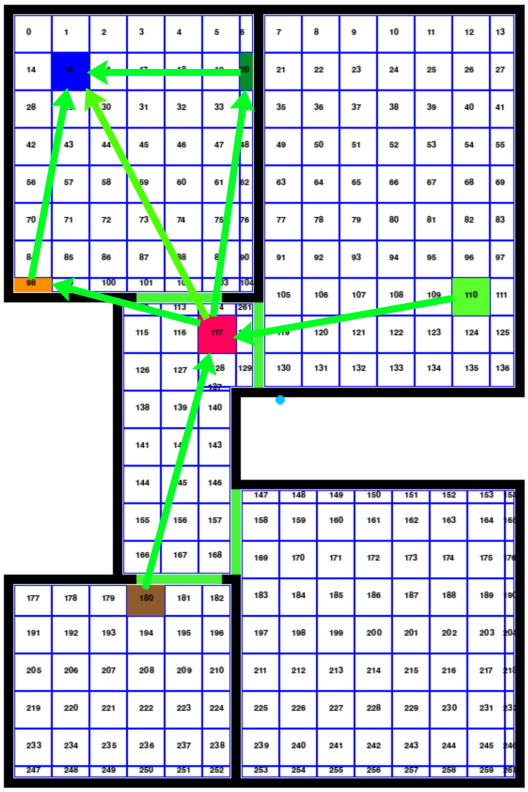


Figure 6.15: Correct way of placing robots during evaluation still performance.

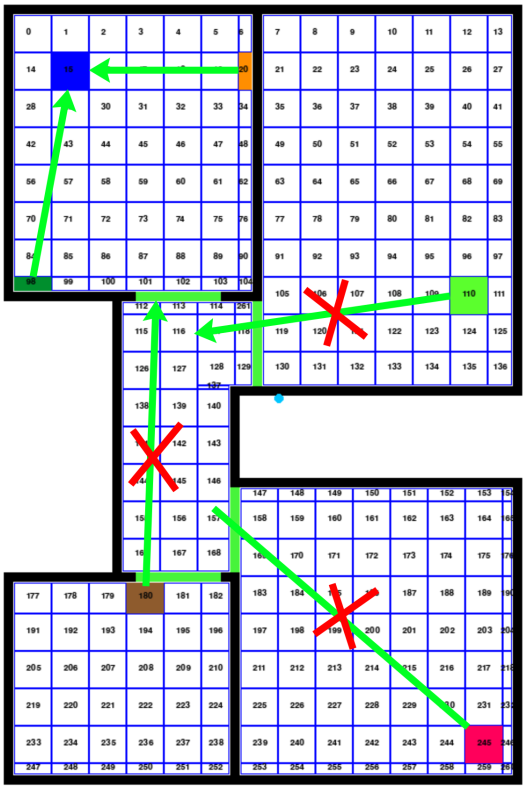


Figure 6.16: Incorrect way of placing robots during evaluation still performance.

Starting with the first scenario: *1) robots are not allowed to move during localization*. During the evaluation of this scenario, the robots are placed at the center of one of the cells on the map, which it will stay in during the entire evaluation. Another requirement here, which was necessary to achieve convergence, is that each robot needs to have at least one LOS connection to another robot with a range below the maximum range (300cm, as derived in section 6.3). This requirement ensures that all robots will contribute to the localization of the robot of interest, which can also be seen in Figure 6.15, where

the position of the six robots are indicated by the colored cells and the green arrows show with which robot each robot can communicate based on the range and obstacles. Figure 6.16 shows the incorrect way of placing the robots on the map, since three of the six robots are not able to communicate with any other robot, as indicated by the red x marks, making them useless when trying the localize the blue robot for example. During the evaluation of the second scenario: *2) robots are allowed to move during localization*, the robots are placed again at the center of a random cell, but without any restrictions. Here, the robots that are out of reach will eventually be able to contact the other robots by driving.

During an evaluation cycle, the first thing done is exchanging messages between the robots, allowing them to process the *hearing another robot* configuration. After all the robots have heard each other, the table sharing will be executed depending on the configuration that is evaluated. During this phase, the table constructed in the previous step will be sent to the robot for which the table was generated. After this, depending on the configuration, the tables are now also shared with the other robots, allowing them to process the *receiving table about other* configuration. Lastly, when the driving scenario is evaluated, the robots start driving across the map, moving exactly one cell. The direction in which they drive is based, just as described in section 5.1, on the particle distribution, where the cell with the highest probability is chosen, and based on the path to that cell the robot is moved towards that cell. This approach was chosen after evaluating multiple driving strategies and it was proven to be the most effective. This evaluation cycle is repeated until the robot of interest is localized by reaching convergence. Convergence is, just as described in section 5.1, achieved when either more than 55% of the particles are in a single cell or the particles are centered around a certain cell with a deviation below 400cm for all particles.

6.6.1. Messages processed

The first metric that was evaluated was the number of messages that were processed by the robot to achieve localization. Here, a processed message is either hearing another robot and constructing a localization table or receiving a localization table from another robot about either self or another robot. As mentioned before, when receiving a localization table from a robot about another robot, also all other configurations are active, and thus more tables are processed in one cycle. During the evaluation, it was discovered that when not allowing the robots to drive, convergence was not always reached resulting in a never-ending cycle. To prevent this, an upper limit of *100 processed messages* was set, after which convergence is assumed and the results are logged. Over the 100 measurements that were taken for each unique situation (active configurations and number of robots), the average number of processed messages is taken.

The results can be seen below in Figure 6.17, for the standing still scenario, and in Figure 6.18, for the driving scenario. Here, it immediately becomes clear that the two different scenarios have the opposite effect on the average number of processed messages, where, while standing still, the number of robots and additional configurations decreases the number of processed messages, while when allowing them to drive it has the opposite effect. This can be explained by the fact that, when standing still, the only information available for the localization is the data received from other robots, which becomes increasingly unique when the number of robots increases, and more configurations are used. So, even though the number of messages processed increases per cycle, the uniqueness of the information allows for faster localization. Another thing that can be observed from these results, is that while driving less messages need to be processed. This shows that allowing the robot to drive positively impacts the localization speed, as it takes less than half of the cycles to reach convergence. Using this knowledge, the opposite effect observed for both scenarios can be explained by the way in which the evaluation is executed, where each cycle has a fixed order, as can be seen in Figure 5.7, of execution ending with the driving of the robot. This means that before the robot is allowed to drive, all other messages are already processed.

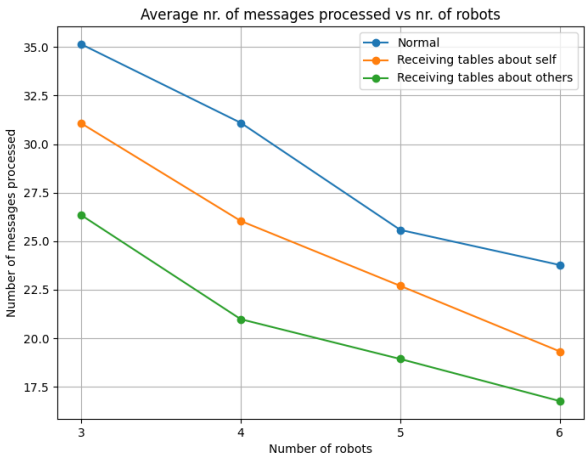


Figure 6.17: Average number of messages processed while standing still.

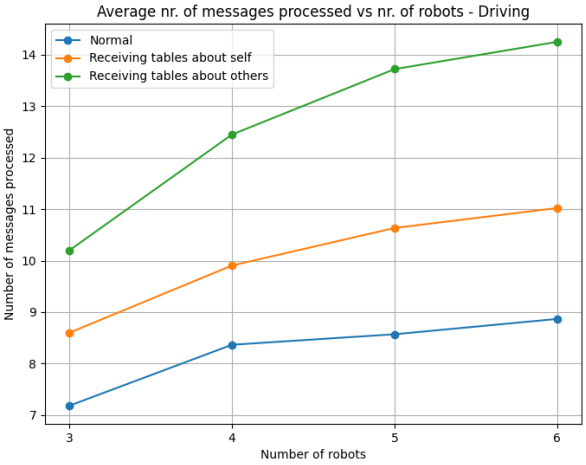


Figure 6.18: Average number of messages processed while driving.

6.6.2. Localization error

Next up, the localization error was evaluated, which also represents the accuracy of the proposed algorithm. To get a complete overview of this part, the localization error was evaluated based on the RMSE, average distance error versus the number of robots, and average distance error versus the number of executed actions. Starting with the RMSE, which is calculated using the same formula, given in Equation 6.2, as used for the evaluation of the particle filter, where the robot's actual position and the robot's estimated position are compared. Here, the RMSE is calculated for every run and logged, so each RMSE gives an overview of the error of that run. After the 100 runs, the average RMSE is calculated, and the results of this can be seen below in Table 6.4. Here, the six unique combinations of standing still or driving and the three configurations each have their own row and for each combination the effect of the number of robots used is given in each column. Based on these results, it can be concluded that using the receiving a table from another robot configuration seems to have a negative impact on the localization accuracy for both the standing still and driving scenario. This is especially true for the driving scenario, where the error becomes *2-5 times* higher when enabling the additional table-sharing configurations. Next, it can also be concluded that the driving scenario performs better than the standing still scenario. Here, in the most ideal situation (only hearing other robots and having six robots) an average RMSE of as low as **3.829 cm** can be achieved, surpassing the performance of the stand-alone particle filter both before and after convergence.

	3 robots	**4 robots**	**5 robots**	**6 robots**
RMSE still - normal	54.101 cm	33.398 cm	20.4255 cm	18.817 cm
RMSE still - receiving self	55.381 cm	36.980 cm	24.709 cm	20.111 cm
RMSE still - receiving other	66.205 cm	52.169 cm	49.768 cm	47.255 cm
RMSE driving - normal	5.954 cm	5.705 cm	4.347 cm	3.829 cm
RMSE driving - receiving self	10.619 cm	10.896 cm	8.905 cm	6.10 cm
RMSE driving - receiving other	27.134 cm	24.766 cm	22.473 cm	19.889 cm

Table 6.4: Proposed algorithm RMSE results.

Next to the RMSE, the final localization error was evaluated against the number of robots. Here, the final localization error is defined as the distance between the estimated position of the robot, calculated using the weighted sum of the particles (Equation 6.3 and Equation 6.4), and the actual position of the robot using the Euclidean distance, which is calculated using Equation 4.4. These results can be seen below in Figure 6.19 for the standing still scenario and in Figure 6.20 for the driving scenario. Here, the same trend as for the RMSE can be observed, where the addition of the table-sharing configurations has a negative impact, and adding more robots has a positive impact on the localization accuracy. Next, allowing the robot to drive seems to increase the algorithm's performance by up to **9.5 times** across

all configurations and used number of robots. The difference in results with respect to the RMSE can be explained by the fact that these plots *only* look at the final error and the RMSE considers the error during each step of the process, where the error can sometimes be lower at earlier steps of the process. Based on the data in Table 6.4, Figure 6.19, and Figure 6.20, the conclusion can be drawn that the scenario in which the robots are allowed to drive is preferred over the scenario in which the robots are not allowed to drive. This scenario allows for cm-level localization precision with errors as low as **13.74cm**, allowing for the right cell to be localized while processing fewer messages than for the standing still scenario. This is in contrast to the standing still scenario, where more messages need to be processed on average and even in the most ideal situation the robot is still localized in a cell next to its true cell.

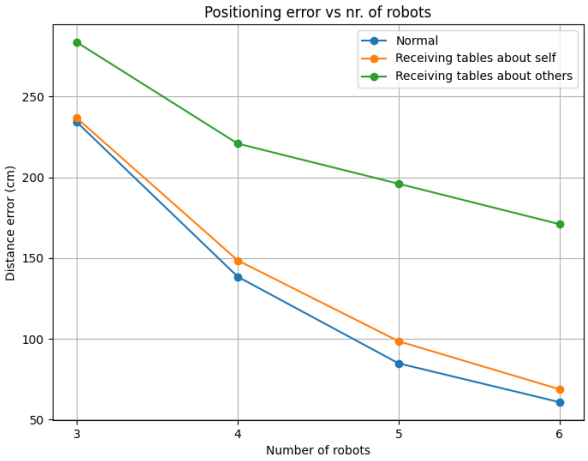


Figure 6.19: Average localization error while standing still.

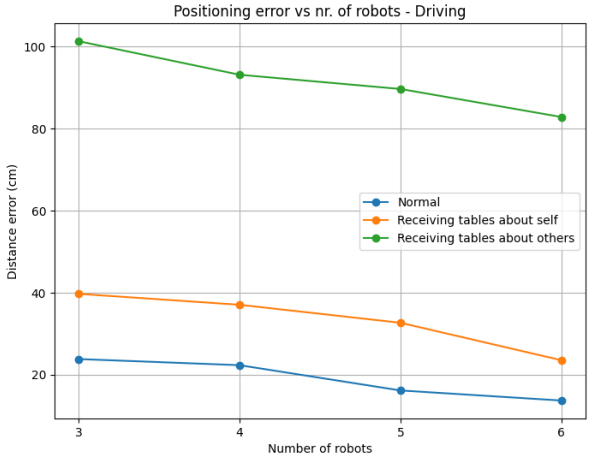
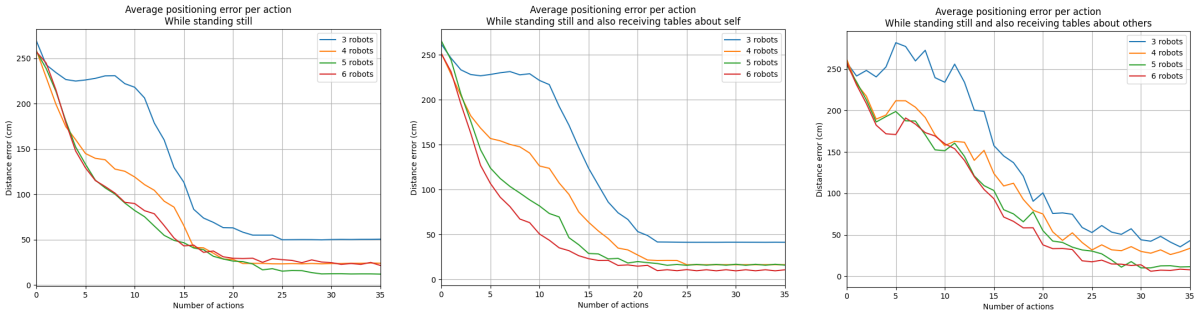


Figure 6.20: Average localization error while driving.

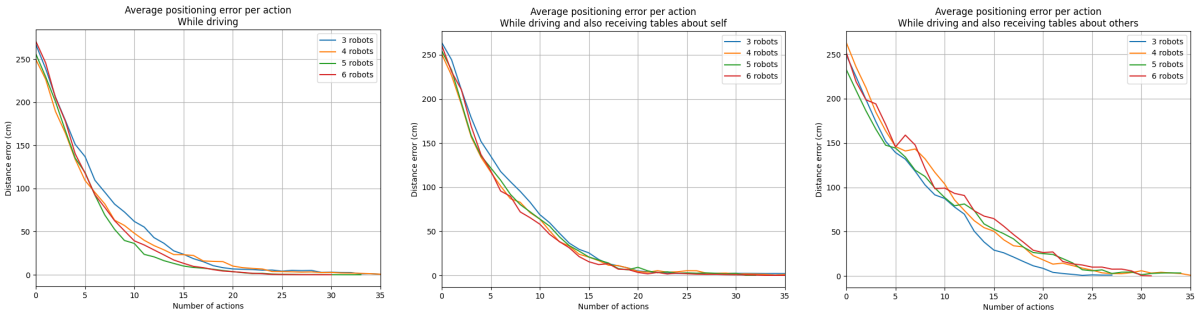
Lastly, the average error with respect to the number of executed actions is evaluated to get an overview of the progression of the localization error during each step of the localization process. Here, an action is defined as either hearing another robot, receiving a table from another robot, or the robot driving. At each of these actions, the localization error is calculated using the *Euclidean distance* between the actual and estimated position of the robot. The results of this evaluation can be seen below in Figure 6.21, where the average localization error versus the number of performed actions is given for each scenario and configuration combination also used in the other parts of the evaluation. Again, the positive impact of allowing the robots to drive can be observed, as the error drops a lot faster in this case compared to the standing still scenario. Next to this, the negative impact of the additional configurations can be observed, as the error tends to drop a little bit slower when the configurations are enabled. In the most ideal situation, it takes *less than 15* actions to achieve a lower average error than 40cm, which allows for the localization in the correct cell. From these graphs, it also becomes clear that the impact of additional robots is mainly visible in the scenario in which the robots are not allowed to move, which can again be explained by the fact that more robots allow for more unique data to be shared underling the robots.

6.6.3. Distance

Based on the evaluation data of the localization accuracy, it was concluded that the scenario where the robots are allowed to drive was preferred over the standing still scenario, as the latter scenario does not allow for the localization in the correct cell. Because of this, the impact of the number of robots and configurations on the traveled distance is evaluated. These results are shown below in Figure 6.22, where the average traveled distance until convergence is given. Here, the positive effect of the additional configurations can be observed, as they allow localization to be achieved while traveling significantly less distance. Next to this, like with all evaluations of this algorithm, it holds that increasing the number of robots positively impacts the results. This positive effect of the table-sharing configurations can be explained by the fact that more unique data is shared between the robots, allowing for better localization estimation while traveling less distance. Based on this, it can be concluded that the choice of whether



(a) Standing still. **(b)** Standing still and receiving tables self. **(c)** Standing still and receiving tables others.



(d) Driving. **(e)** Driving and receiving tables self. **(f)** Driving and receiving tables others.

Figure 6.21: Localization error vs the number of executed actions.

to use the additional configurations depends on a trade-off between the localization accuracy and the traveled distance. In the best case, the algorithm can achieve convergence while only driving **363 cm**. Next to this, it can also be concluded the only two useful combinations are *driving and only hearing other robots* and *driving and receiving table about self from other robots*, as these two combinations both allow the robot to be localized in the correct cell. Also, receiving tables about other robots would significantly lower the travel distance, but this will come at the cost of localization accuracy, making it impossible to localize the robot in the correct cell. Here, the proposed algorithm again outperforms the stand-alone particle filter, where even the worst-case distance of *550cm* outperforms the best-case distance of *628.32cm* of the normal particle filter.

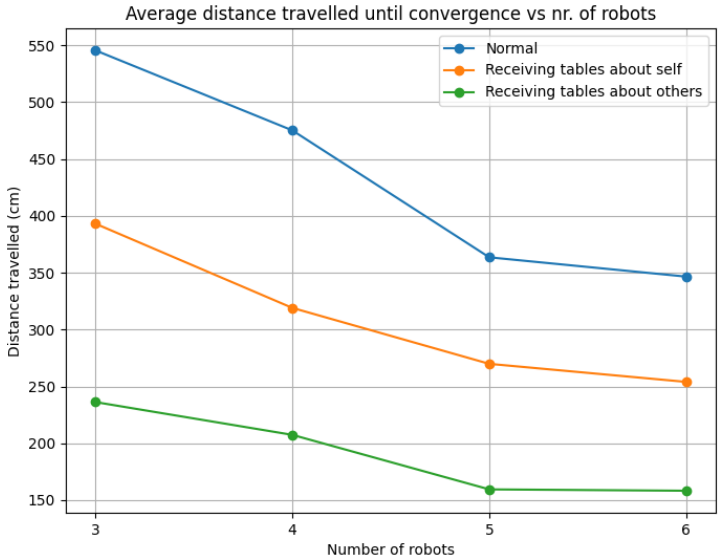


Figure 6.22: Average distance traveled to localize the robot.

6.6.4. Localization speed

For the last part of the evaluation of the proposed algorithm, the localization speed was evaluated, which indicates how fast convergence is achieved in seconds when starting from a uniform distribution. Based on the evaluation results, given in subsection 6.6.1 and subsection 6.6.2, it was concluded that the only two valid combinations are the case in which the robot is driving and only hearing other robots and the case in which the robot is driving and receiving tables about himself from other robots. Because of this, these two combinations are the only ones for which the localization speed was calculated. To calculate the localization time, Equation 6.5 is used, where $t_{localization}$ is the average localization time in seconds, $t_{message}$ is the time it takes to process a single message in seconds, $n_{message}$ is the average number of messages processed, and $t_{driving}$ is the time spent driving in seconds. Using this equation, it was calculated that the localization time, for the ideal situation of having six active robots, is **9.252s** when only hearing other robots and **10.403s** when also receiving tables from other robots.

$$t_{localization} = t_{message} * n_{message} + t_{driving} \tag{6.5}$$

Based on this, for this category the proposed algorithm does not necessarily outperform the normal particle filter, as the localization time is *2-3 times* higher. This, however, can be explained by the fact that sending a message takes time, especially at a hardware-limited sampling rate of 44.1kHz. The localization speed could thus be improved by using different hardware that allows a higher sampling rate, allowing faster data transfer and faster localization results. The proposed algorithm does, however, outperform in localization accuracy and traveled distance, allowing for more accurate localization results while not driving in the wrong direction for a long time. Next to this, when the robot would get locked in a room, this algorithm would still be able to localize the robot while the particle filter might not be able to. Looking at all results, a clear winner between the two combinations cannot be chosen, since it is a trade-off between the localization accuracy and traveled distance, whose importance is based on the situation. Because of this, the option to enable the sharing of tables between robots is added to the configuration file, allowing for easy switching between both combinations based on the requirements of the situation and environment.

6.7. Comparison to other work

After gathering all the evaluation results of the proposed work of this thesis and all the separate parts used to make the algorithm possible, the last thing that needs to be done to complete this thesis work is comparing the results to other similar works. This comparison to other works will show the academic significance of the research executed in this thesis. Since the proposed work in this thesis, at the moment of writing, is of a unique nature, it was quite hard to find work comparable to this one. Because of this, similar works were selected based on the criteria that they at least use SSL to achieve indoor localization results. Next to this, the performance of the normal particle filter was also added to show how the proposed algorithm improves on that. The results of this comparison can be seen below in Table 6.5.

Here, the works marked with a (*) do not actually determine the robot's position inside the building, but rather determine the robot's location relative to another robot in the swarm. Unfortunately, not all data was available in the research papers of the compared works and because of this, these results are left blank in the table. For some of the other data that is present in the table, the data had to be deduced from plots that were available in the paper. Because of this, these results are not completely accurate and rounded up to cm precision. For all the data available in the papers, the best-case scenarios were chosen to be used in the comparison table for the case where multiple scenarios and results were described. The same was done for the proposed work in this thesis, making the comparison as accurate as possible. The localization accuracy column in the table represents the final localization error after completing the localization process.

	Localization accuracy	RMSE	Localization speed	Localization distance	Data rate	Noise resilience
Sonic Filter Localization	13.74cm	3.83cm	9.25s	350cm	1.302bps/kHz	-12dB
Normal PF	15.96cm	256.67cm	3.2s	628.32cm	-	-
AudioLocNet [28]*	44cm	-	-	-	1.206bps/kHz	-12dB
Bluetooth floor-plan [29]	15cm	50cm	-	-	-	-
AALTS [123]	49cm	-	2s	-	-	-1dB
Indoor pseudo-ranging [124]	20cm	-	-	-	33.3bps/kHz	1.4dB
Guoguo [125]	10cm	8cm	60s	-	-	-
RAIPS [126]	50cm	-	-	-	-	-
ASSIST [127]	30cm	-	-	45m	-	-

Table 6.5: Comparison to other works.

Looking at the results of the comparison given in Table 6.5, it can clearly be seen that the proposed algorithm of this thesis outperforms most of the similar works. Only one of the works can achieve a better localization accuracy, but when looking at the RMSE value and localization speed of this work, the proposed algorithm again outperforms it in these categories. This shows that the proposed algorithm can achieve a more accurate clue of the robot's position faster than the other algorithms. Next to this, the localization speed of the proposed algorithm is not the fastest among all other algorithms. This can be explained by the fact that the data rate of the algorithm is quite low, resulting in the fact that sharing data between robots takes time. Because the proposed algorithm is based on the fact that robots share information over audio and the data rate is limited by the hardware used, it is impossible to increase the localization time. Future works could achieve faster localization results by using better hardware, including a microphone array that supports a higher sampling rate, allowing for a higher data rate.

Looking at the data rates of the compared works, the proposed algorithm is only outperformed by one other algorithm. This can be explained by looking at the noise resilience of the two algorithms, where the algorithm with the higher data rate has a much lower noise resilience. For the proposed algorithm, noise resilience was chosen over the data rate, as correctly receiving the messages was deemed more important than the speed with which this is done. Another important thing that needs to be noted is that all other works in this list use either some form of AI or beacons to achieve indoor localization. This means that these works cannot be deployed in every environment, as either training or adapting the environment will always be required for the algorithms to work. This is in contrast to the work described in this thesis, which only needs a map of the building to be able to localize the robot in an indoor environment. Since most buildings already have a construction plan, a software tool could be written which would be able to translate such plans into the correct format. This allows the proposed algorithm to work anywhere without the need for changes to the environment or excessive preparation.

7

Conclusion

The research described in this thesis is aimed at achieving high accuracy indoor localization for robotic car swarms. Here, the main research question of this work is stated as follows: 'How can a particle filter and sound source localization techniques be combined to achieve indoor localization based on a map?'. To answer this question, a small microphone array consisting of six microphones was placed on top of a robotic car. Together with this microphone array, a robust SSL algorithm was developed, which allowed the robot to determine the DOA and distance from the messages it receives. However, the DOA and distance are not accurate enough on their own to deduce the location of the robot. To overcome these inaccuracies, a novel approach was invented, that combines the existing techniques of a particle filter and SSL. What makes this novel approach so unique is that, in contrast to most of the other works in the field of indoor sound-based localization, it does *not require* any form of beacons to be pre-installed inside the building and it does *not make use of* any form of artificial intelligence. Instead, this new approach makes use of probability theory, where based on the deduced DOA and distance of a received message the robot tries to estimate from which cell the message could have originated. This is done by generating so-called *localization tables*, which depict the possible cell combinations, which can then be used to update the positions of the particles on the map. Combining this with the ability to drive, allows the robot to localize itself based on the weight and coordinates of the particles on the map.

For the proposed algorithm, a distinction can be made between three scenarios: 1) Hearing another robot, 2) Receiving a localization table about self, and 3) Receiving a localization table about another robot. Next to this, another distinction can be made between a scenario in which the robot is standing still during the localization process or when it's driving. During the evaluation stage, the performance of the SSL was evaluated under LOS, NLOS, and reverberating conditions, covering most of the situations in which a robotic car could find itself while navigating an indoor environment. During the evaluation of the proposed algorithm, it was discovered that allowing the robots to drive resulted in the best performance out of the two scenarios, allowing for cell-level precision during the localization. During the evaluation of the three scenarios of the algorithm, it was discovered that using the more advanced (3) scenario resulted in less distance needed to be traveled before achieving convergence, while using the less advanced (1) scenario resulted in better localization accuracy. Next to this, it was discovered that deploying more robots also leads to better performance, regardless of the chosen scenario. After completing the evaluation, it was concluded that the use of scenario 1 in combination with driving resulted in the best system performance, where the number of exchanged messages is kept to a minimum and an RMSE of as low as **3.83 cm** could be achieved. Here, the robot only needs to drive **363 cm** on average to localize itself, making sure that accidental driving in the wrong direction is kept to a minimum. In addition to this, the total localization time of the robot is only **9.52 seconds**. These results outperform most of the similar works in the field of indoor localization using sound-based signals [28, 29, 123–127], as it can achieve higher accuracy localization results while exchanging a minimal number of messages and driving a minimal amount of distance.

Based on the obtained results, it can be concluded that the research question has been thoroughly answered by the proposed algorithm. The proposed algorithm combines a particle filter and SSL in a novel approach using probability theory to achieve high accuracy indoor localization results. Here, it can also be concluded that the choice for using audio-based signals in this work proved to be an excellent one. Its unique characteristics, such as limited range, not passing through walls, and bending around walls, in combination with a detailed map of the environment allowed for an accurate generation of the localization table, which lies at the heart of the algorithm. The research that has been done in this thesis will serve as a great contribution to the field of indoor localization using sound-based signals, demonstrating that sound, because of its unique qualities, is a solid and accurate choice of technique to be used when localizing an object of interest without the need of structural changes or the use of machine learning.

The work in this thesis has some minor limitations, which are mostly caused by the limitations of the chosen hardware. Further research, using specialized hardware, into the proposed algorithm could provide even more insights and allow the work to bring an even bigger contribution to the field of indoor localization using sound signals. Future works could, for example, solve the hardware limitation by designing a custom microphone array with microphones capable of sampling at higher rates. This would allow research into the effect of deploying more than six robots in the swarm and determine the optimal number of robots to deploy on a large scale. Next to this, solving the hardware issue would allow for higher data rates, resulting in faster localization results. Another interesting aspect that would benefit from further research would be implementing the whole program on a single robotic car board, eliminating the need for the power-hungry Raspberry Pi and allowing research into the efficiency and power consumption of the algorithm. Lastly, the impact of letting robots communicate simultaneously during localization would be an interesting aspect to investigate further. Due to the way the localization cycle is currently set up, it is prevented that multiple robots communicate at the same time as the simultaneous communication was considered out of the scope of this research. Altering this cycle to let the robots communicate simultaneously is possible since the messaging codec already supports it and it would allow for faster data transfer, which in turn could result in the faster localization of the robot.

References

[1] Erol Şahin. “Swarm Robotics: From Sources of Inspiration to Domains of Application”. In: *Swarm Robotics*. Ed. by Erol Şahin and William M. Spears. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 10–20. ISBN: 978-3-540-30552-1.

[2] Simon Garnier, Jacques Gautrais, and Guy Theraulaz. “The biological principles of swarm intelligence”. In: *Swarm Intelligence* 1.1 (June 2007), pp. 3–31. ISSN: 1935-3820. DOI: 10.1007/s11721-007-0004-y.

[3] Yuko Ulrich et al. “Emergent behavioral organization in heterogeneous groups of a social insect”. In: *bioRxiv* (2020). DOI: 10.1101/2020.03.05.963207.

[4] Nicholas Claggett et al. “Termite Mounds: Bioinspired Examination of the Role of Material and Environment in Multifunctional Structural Forms”. In: *Journal of Structural Engineering* 144.7 (2018), p. 02518001. DOI: 10.1061/(ASCE)ST.1943-541X.0002043.

[5] Octavio Miramontes and Og DeSouza. “Individual Basis for Collective Behaviour in the Termite, Cornitermes cumulans”. In: *Journal of insect science* 8.22 (2008), pp. 1–11. DOI: 10.1673/031.008.2201.

[6] A. R. Mermut, M. A. Arshad, and R. J. St. Arnaud. “Micropedological Study of Termite Mounds of Three Species of Macrotermes in Kenya”. In: *Soil Science Society of America Journal* 48.3 (1984), pp. 613–620. DOI: https://doi.org/10.2136/sssaj1984.03615995004800030029x.

[7] Fethi Matoui et al. “Contribution to the path planning of a multi-robot system: centralized architecture”. In: *Intelligent Service Robotics* 13.1 (Jan. 2020), pp. 147–158. ISSN: 1861-2784. DOI: 10.1007/s11370-019-00302-w.

[8] Iñaki Navarro and Fernando Matía. “An Introduction to Swarm Robotics”. In: *ISRN Robotics* 2013 (Sept. 2012), p. 608164. ISSN: 2356-7872. DOI: 10.5402/2013/608164.

[9] Emaad Mohamed H. Zahugi, Ahmed M. Shabani, and T. V. Prasad. “Libot: Design of a low cost mobile robot for outdoor swarm robotics”. In: *2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. 2012, pp. 342–347. DOI: 10.1109/CYBER.2012.6392577.

[10] G. Vásárhelyi et al. “Outdoor flocking and formation flight with autonomous aerial robots”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 3866–3873. DOI: 10.1109/IROS.2014.6943105.

[11] Siyuan Chen, Dong Yin, and Yifeng Niu. “A Survey of Robot Swarms & Relative Localization Method”. In: *Sensors* 22.12 (2022). ISSN: 1424-8220. DOI: 10.3390/s22124424.

[12] Ehsan Latif and Ramviyas Parasuraman. “Online Indoor Localization Using DOA of Wireless Signals”. In: *CoRR* abs/2201.05105 (2022). arXiv: 2201.05105. URL: https://arxiv.org/abs/2201.05105.

[13] Stefano Rinaldi et al. “An Evaluation of Low-Cost Self-Localization Service Exploiting Angle of Arrival for Industrial Cyber-Physical Systems”. In: *2021 IEEE AFRICON*. 2021, pp. 1–6. DOI: 10.1109/AFRICON51333.2021.9570985.

[14] Yu Xianjia et al. “Applications of UWB Networks and Positioning to Autonomous Robots and Industrial Systems”. In: *2021 10th Mediterranean Conference on Embedded Computing (MECO)*. 2021, pp. 1–6. DOI: 10.1109/MECO52532.2021.9460266.

[15] Paola Torrico Morón, Jorge Peña Queralta, and Tomi Westerlund. “Towards Large-Scale Relative Localization in Multi-Robot Systems with Dynamic UWB Role Allocation”. In: *2022 7th International Conference on Robotics and Automation Engineering (ICRAE)*. 2022, pp. 239–246. DOI: 10.1109/ICRAE56463.2022.10054614.

[16] Lun Hai et al. “An improved weighted centroid localization algorithm based on Zigbee”. In: *2022 5th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*. 2022, pp. 634–637. DOI: 10.1109/AEMCSE55572.2022.00129.

[17] Mary Ann E. Latina, Alleah Reyes, and Edzel Marius Rollon. “Optimization of RSSI-based Zigbee Indoor Localization System for Determining Distances Between Unknown Nodes”. In: *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*. 2022, pp. 1–6. DOI: 10.1109/ICEEICT53079.2022.9768601.

[18] Emidio DiGiampaolo, Francesco Martinelli, and Fabrizio Romanelli. “Exploiting the Orientation of Trilateration UHF RFID Tags in Robot Localization and Mapping”. In: *2022 IEEE 12th International Conference on RFID Technology and Applications (RFID-TA)*. 2022, pp. 5–8. DOI: 10.1109/RFID-TA54958.2022.9924022.

[19] Fabio Bernardini et al. “Retail Robots with UHF-RFID Moving Antennas enabling 3D Localization”. In: *2022 IEEE 12th International Conference on RFID Technology and Applications (RFID-TA)*. 2022, pp. 1–4. DOI: 10.1109/RFID-TA54958.2022.9924120.

[20] Hande Celikkanat. “Optimization of felf-organized flocking of a robot swarm via evolutionary strategies”. In: *2008 23rd International Symposium on Computer and Information Sciences*. 2008, pp. 1–4. DOI: 10.1109/ISCIS.2008.4717880.

[21] Oscar De Silva, George K. I. Mann, and Raymond G. Gosine. “Development of a relative localization scheme for ground-aerial multi-robot systems”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 870–875. DOI: 10.1109/IROS.2012.6386015.

[22] A. Korian, Shivam Sharma, and V. K. Mittal. “Wireless audio communication over Infra-Red medium”. In: *2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN)*. 2016, pp. 240–245. DOI: 10.1109/SPIN.2016.7566696.

[23] Burak Soner and Sinem Coleri. “Visible Light Communication Based Vehicle Localization for Collision Avoidance and Platooning”. In: *IEEE Transactions on Vehicular Technology* 70.3 (2021), pp. 2167–2180. DOI: 10.1109/TVT.2021.3061512.

[24] Isamu Takai et al. “LED and CMOS Image Sensor Based Optical Wireless Communication System for Automotive Applications”. In: *IEEE Photonics Journal* 5.5 (2013), pp. 6801418–6801418. DOI: 10.1109/JPHOT.2013.2277881.

[25] Alexander B. Maxseiner, Daniel M. Lofaro, and Donald A. Sofge. “Visible Light Communications with Inherent Agent Localization and Simultaneous Message Receiving Capabilities for Robotic Swarms”. In: *2021 18th International Conference on Ubiquitous Robots (UR)*. 2021, pp. 633–639. DOI: 10.1109/UR52253.2021.9494636.

[26] Chenming Xin et al. “Research on Indoor Navigation System of UAV Based on LIDAR”. In: *2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*. 2020, pp. 763–766. DOI: 10.1109/ICMTMA50254.2020.00166.

[27] Ningbo Li, Lianwu Guan, and Yanbin Gao. “A Seamless Indoor and Outdoor Low-cost Integrated Navigation System Based on LIDAR/GPS/INS”. In: *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. 2020, pp. 1–6. DOI: 10.1109/VTC2020-Fall49728.2020.9348869.

[28] Amjad Yousef Mjaid et al. “AI-based Simultaneous Audio Localization and Communication for Robots”. In: IoTDI ’23. , San Antonio, TX, USA, Association for Computing Machinery, 2023, pp. 172–183. DOI: 10.1145/3576842.3582373.

[29] Shihao Xu et al. “Bluetooth, Floor-Plan, and Microelectromechanical Systems-Assisted Wide-Area Audio Indoor Localization System: Apply to Smartphones”. In: *IEEE Transactions on Industrial Electronics* 69.11 (2022), pp. 11744–11754. DOI: 10.1109/TIE.2021.3111561.

[30] Ish Rishabh, Don Kimber, and John Adcock. “Indoor localization using controlled ambient sounds”. In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2012, pp. 1–10. DOI: 10.1109/IPIN.2012.6418905.

[31] Oriol Vinyals, Eladio Martin, and Gerald Friedland. “Multimodal Indoor Localization: An Audio-Wireless-Based Approach”. In: *2010 IEEE Fourth International Conference on Semantic Computing*. 2010, pp. 120–125. DOI: 10.1109/ICSC.2010.87.

[32] Yunqiang Chen and Yong Rui. “Real-time speaker tracking using particle filter sensor fusion”. In: *Proceedings of the IEEE* 92.3 (2004), pp. 485–494. DOI: 10.1109/JPROC.2003.823146.

[33] Jean-Samuel Lauzon et al. “Localization of RW-UAVs using particle filtering over distributed microphone arrays”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 2479–2484. DOI: 10.1109/IROS.2017.8206065.

[34] Sangwon Kim, Jimi Lee, and Byoung Chul Ko. “SSL-MOT: self-supervised learning based multi-object tracking”. In: *Applied Intelligence* 53.1 (Jan. 2023), pp. 930–940. ISSN: 1573-7497. DOI: 10.1007/s10489-022-03473-9.

[35] Andrew Pyzdek. “The World Through Sound: Wavelength”. In: (2024). URL: https://acousticstoday.org/wavelength/#:~:text=Namely%2C%20if%20we%20know%20the,equation%20wavelength%3Dspeed%2Ffrequency..

[36] Allan Pierce. “Diffraction of sound around corners and over wide barriers”. In: *Journal of The Acoustical Society of America - J ACOUST SOC AMER* 55 (May 1974). DOI: 10.1121/1.1914668.

[37] Jun Yin and Marian Verhelst. “CNN-based Robust Sound Source Localization with SRP-PHAT for the Extreme Edge”. In: *ACM Trans. Embed. Comput. Syst.* 22.3 (Apr. 2023). ISSN: 1539-9087. DOI: 10.1145/3586996.

[38] Fucai Hu et al. “Sound source localization based on residual network and channel attention module.” In: vol. 13. 1. 2023, p. 5443. DOI: https://doi.org/10.1038/s41598-023-32657-7.

[39] Jun Tang et al. “Sound source localization method based time-domain signal feature using deep learning”. In: *Applied Acoustics* 213 (2023), p. 109626. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2023.109626.

[40] Tan-Hsu Tan et al. “Sound Source Localization Using a Convolutional Neural Network and Regression Model”. In: *Sensors* 21.23 (2021). ISSN: 1424-8220. DOI: 10.3390/s21238031.

[41] Soo Young Lee, Jiho Chang, and Seungchul Lee. “Deep learning-based method for multiple sound source localization with high resolution and accuracy”. In: *Mechanical Systems and Signal Processing* 161 (2021), p. 107959. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2021.107959.

[42] Luoyi Feng et al. “A double-step grid-free method for sound source identification using deep learning”. In: *Applied Acoustics* 201 (2022), p. 109099. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2022.109099.

[43] Paolo Castellini et al. “A neural network based microphone array approach to grid-less noise source localization”. In: *Applied Acoustics* 177 (2021), p. 107947. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2021.107947.

[44] Qingbo Zhai et al. “A grid-free global optimization algorithm for sound sources localization in three-dimensional reverberant environments”. In: *Mechanical Systems and Signal Processing* 188 (2023), p. 109999. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2022.109999.

[45] W. Ma and X Liu. “Phased microphone array for sound source localization with deep learning.” In: vol. 2. 2. 2019, pp. 71–81. DOI: https://doi.org/10.1007/s42401-019-00026-w.

[46] Sebastian Thrun. “Particle filters in robotics”. In: *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*. UAI’02. Alberta, Canada: Morgan Kaufmann Publishers Inc., 2002, pp. 511–518. ISBN: 1558608974.

[47] Samira Hayat, Evşen Yanmaz, and Raheeb Muzaffar. “Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint”. In: *IEEE Communications Surveys & Tutorials* 18.4 (2016), pp. 2624–2661. DOI: 10.1109/COMST.2016.2560343.

[48] Guang-Zhong Yang et al. “The grand challenges of Science Robotics”. In: *Science Robotics* 3.14 (2018), eaar7650. DOI: 10.1126/scirobotics.aar7650.

[49] P. Bahl and V.N. Padmanabhan. “RADAR: an in-building RF-based user location and tracking system”. In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*. Vol. 2. 2000, 775–784 vol.2. DOI: 10.1109/INFCOM.2000.832252.

[50] Salam Alyafawi. “Real-Time Localization using Software Defined Radio”. PhD thesis. Aug. 2015.

[51] *2023 Bluetooth® Market Update*. 2023. URL: https://www.bluetooth.com/2023-market-update/ (visited on 11/08/2023).

[52] *Understanding Bluetooth Range*. 2023. URL: https://www.bluetooth.com/learn-about-bluetooth/key-attributes/range/#:~:text=Bluetooth%C2%AE%20technology%20uses%20the,balance%20between%20range%20and%20throughput. (visited on 11/08/2023).

[53] Zhu Jianyong et al. “RSSI based Bluetooth low energy indoor positioning”. In: *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2014, pp. 526–533. DOI: 10.1109/IPIN.2014.7275525.

[54] *Bluetooth® Core Specification Version 5.1 Feature Overview*. 2019. URL: https://www.bluetooth.com/bluetooth-resources/bluetooth-core-specification-v5-1-feature-overview/ (visited on 11/08/2023).

[55] *Bluetooth Direction Finding: A Technical Overview*. 2019. URL: https://www.bluetooth.com/bluetooth-resources/bluetooth-direction-finding/ (visited on 11/08/2023).

[56] *Adopting Ultra-Wideband for Memsen’s file sharing and wireless marketing platform*. 2023. URL: https://www.fcc.gov/file/14377/download (visited on 11/08/2023).

[57] Bernhard Großiwindhager et al. “SnapLoc: An Ultra-Fast UWB-Based Indoor Localization System for an Unlimited Number of Tags”. In: *2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 2019, pp. 61–72.

[58] Xiaosi Chen et al. “Network scalability with weight analysis based on UWB indoor positioning system”. In: *2016 IEEE Conference on Wireless Sensors (ICWiSE)*. 2016, pp. 95–99. DOI: 10.1109/ICWISE.2016.8188549.

[59] Zigbee Alliance. *Zigbee Specification Version 1.0*. https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf. Accessed on 09-11-2023. 2004.

[60] Sinem Coleri Ergen. *ZigBee/IEEE 802.15.4 Summary*. http://users.eecs.northwestern.edu/~peters/references/ZigtbeeIEEE802.pdf. Accessed on 09-11-2023. 2004.

[61] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. “A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi”. In: *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*. 2007, pp. 46–51. DOI: 10.1109/IECON.2007.4460126.

[62] Alp Ustundag and Mehmet Tanyas. “The impacts of Radio Frequency Identification (RFID) technology on supply chain costs”. In: *Transportation Research Part E: Logistics and Transportation Review* 45.1 (2009), pp. 29–38. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2008.09.001. URL: https://www.sciencedirect.com/science/article/pii/S1366554508001154.

[63] Giorgia Casella, Barbara Bigliardi, and Eleonora Bottani. “The evolution of RFID technology in the logistics field: a review”. In: *Procedia Computer Science* 200 (2022). 3rd International Conference on Industry 4.0 and Smart Manufacturing, pp. 1582–1592. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.01.359. URL: https://www.sciencedirect.com/science/article/pii/S1877050922003684.

[64] Stephen Reutebuch, Hans-Erik Andersen, and Robert Mcgaughey. “Light Detection and Ranging (LIDAR): An Emerging Tool for Multiple Resource Inventory”. In: *Journal of Forestry* 103 (Sept. 2005), pp. 286–292.

[65] Jidong Feng et al. “Novel LiDAR-assisted UWB positioning compensation for indoor robot localization”. In: *2021 International Conference on Advanced Mechatronic Systems (ICAMechS)*. 2021, pp. 215–219. DOI: 10.1109/ICAMechS54019.2021.9661496.

[66] H. Durrant-Whyte and T. Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE Robotics & Automation Magazine* 13.2 (2006), pp. 99–110. DOI: 10.1109/MRA.2006.1638022.

[67] Caleb Rascon and Ivan Meza. “Localization of sound sources in robotics: A review”. In: *Robotics and Autonomous Systems* 96 (2017), pp. 184–210. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2017.07.011. URL: https://www.sciencedirect.com/science/article/pii/S0921889016304742.

[68] L. A. Jeffress. “A place theory of sound localization”. In: *Journal of Comparative and Physiological Psychology* 41 (1948), pp. 35–39. DOI: https://doi.org/10.1037/h0061495.

[69] Shikha Thakur and Sneha Singh. “Sound source localization of harmonic sources in entire 3D space using just 5 acoustic signals”. In: *Applied Acoustics* 201 (2022), p. 109126. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2022.109126. URL: https://www.sciencedirect.com/science/article/pii/S0003682X2200500X.

[70] Shuai Cao et al. “Effective Audio Signal Arrival Time Detection Algorithm for Realization of Robust Acoustic Indoor Positioning”. In: *IEEE Transactions on Instrumentation and Measurement* 69.10 (2020), pp. 7341–7352. DOI: 10.1109/TIM.2020.2981985.

[71] Ruizhi Chen et al. “Precise Indoor Positioning Based on Acoustic Ranging in Smartphone”. In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–12. DOI: 10.1109/TIM.2021.3082269.

[72] Guanqun Liu et al. “A Sound Source Localization Method Based on Microphone Array for Mobile Robot”. In: *2018 Chinese Automation Congress (CAC)*. 2018, pp. 1621–1625. DOI: 10.1109/CAC.2018.8623702.

[73] M. M. Saad et al. “High-Accuracy Reference-Free Ultrasonic Location Estimation”. In: *IEEE Transactions on Instrumentation and Measurement* 61.6 (2012), pp. 1561–1570. DOI: 10.1109/TIM.2011.2181911.

[74] Luka Kraljević et al. “Free-Field TDOA-AOA Sound Source Localization Using Three Soundfield Microphones”. In: *IEEE Access* 8 (2020), pp. 87749–87761. DOI: 10.1109/ACCESS.2020.2993076.

[75] Chinmayi Mahapatra and A.R. Mohanty. “Explosive sound source localization in indoor and outdoor environments using modified Levenberg Marquardt algorithm”. In: *Measurement* 187 (2022), p. 110362. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2021.110362. URL: https://www.sciencedirect.com/science/article/pii/S0263224121012562.

[76] De-Bing Zhuo and Hui Cao. “Fast Sound Source Localization Based on SRP-PHAT Using Density Peaks Clustering”. In: *Applied Sciences* 11.1 (2021). ISSN: 2076-3417. DOI: 10.3390/app11010445. URL: https://www.mdpi.com/2076-3417/11/1/445.

[77] Norikazu Ikoma et al. “Tracking of 3D sound source location by particle filter with TDOA and signal power ratio”. In: *2009 ICCAS-SICE*. 2009, pp. 1374–1377.

[78] Jing Fan, Qian Luo, and Ding Ma. “Localization estimation of sound source by microphones array”. In: *Procedia Engineering* 7 (2010). 2010 Symposium on Security Detection and Information Processing, pp. 312–317. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2010.11.050. URL: https://www.sciencedirect.com/science/article/pii/S1877705810010441.

[79] Wuyang Jiang et al. “Two-stage Localisation Scheme Using a Small-scale Linear Microphone Array for Indoor Environments”. In: *Journal of Navigation* 68.5 (2015), pp. 915–936. DOI: 10.1017/S0373463315000107.

[80] Hong W. Paik JW Lee JH. “An Enhanced Smoothed L0-Norm Direction of Arrival Estimation Method Using Covariance Matrix”. In: *Sensors (Basel).* 21.13 (2021). DOI: doi:10.3390/s21134403.

[81] Jacek P. Dmochowski, Jacob Benesty, and Sofiene Affes. “Broadband Music: Opportunities and Challenges for Multiple Source Localization”. In: *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 2007, pp. 18–21. DOI: 10.1109/ASPAA.2007.4392978.

[82] Nabeel Lafta and Saad Hreshee. “Wireless sensor network’s localization based on multiple signal classification algorithm”. In: *International Journal of Electrical and Computer Engineering (IJECE)* 11 (Feb. 2021), p. 498. DOI: 10.11591/ijece.v11i1.pp498-507.

[83] Hoang Do, Harvey F. Silverman, and Ying Yu. “A Real-Time SRP-PHAT Source Location Implementation using Stochastic Region Contraction(SRC) on a Large-Aperture Microphone Array”. In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP ’07*. Vol. 1. 2007, pp. I-121-I–124. DOI: 10.1109/ICASSP.2007.366631.

[84] J.-M. Valin et al. “Localization of simultaneous moving sound sources for mobile robot using a frequency- domain steered beamformer approach”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*. Vol. 1. 2004, 1033–1038 Vol.1. DOI: 10.1109/ROBOT.2004.1307286.

[85] Sarthak Khanal and Harvey F. Silverman. “Multi-stage rejection sampling (MSRS): A robust SRP-PHAT peak detection algorithm for localization of cocktail-party talkers”. In: *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 2015, pp. 1–5. DOI: 10.1109/WASPAA.2015.7336887.

[86] Jacek P. Dmochowski, Jacob Benesty, and SofiÈne Affes. “A Generalized Steered Response Power Method for Computationally Viable Source Localization”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.8 (2007), pp. 2510–2526. DOI: 10.1109/TASL.2007.906694.

[87] Hamid R. Zarghi, Mohamad Sharifkhani, and Iman Gholampour. “Implementation of a cost efficient SSL based on an Angular beamformer SRP-PHAT”. In: *2011 18th IEEE International Conference on Electronics, Circuits, and Systems*. 2011, pp. 49–52. DOI: 10.1109/ICECS.2011.6122211.

[88] *The state of AI in 2022—and a half decade in review*. 2022. URL: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2022-and-a-half-decade-in-review (visited on 01/26/2024).

[89] Gary W Siebein. “Understanding Classroom Acoustic Solutions”. In: *Semin Hear* 25.2 (2004), pp. 141–154. DOI: 10.1055/s-2004-828665.

[90] P.M. Djuric et al. “Particle filtering”. In: *IEEE Signal Processing Magazine* 20.5 (2003), pp. 19–38. DOI: 10.1109/MSP.2003.1236770.

[91] F. Gustafsson et al. “Particle filters for positioning, navigation, and tracking”. In: *IEEE Transactions on Signal Processing* 50.2 (2002), pp. 425–437. DOI: 10.1109/78.978396.

[92] Sebastian Thrun. “Particle Filters in Robotics.” In: *UAI*. Vol. 2. Citeseer. 2002, pp. 511–518.

[93] John H. Halton. “A Retrospective and Prospective Survey of the Monte Carlo Method”. In: *SIAM Review* 12.1 (1970), pp. 1–63. ISSN: 00361445. URL: http://www.jstor.org/stable/2029039 (visited on 04/13/2024).

[94] J.-M. Valin, F. Michaud, and J. Rouat. “Robust 3D Localization and Tracking of Sound Sources Using Beamforming and Particle Filtering”. In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. Vol. 4. 2006, pp. IV–IV. DOI: 10.1109/ICASSP.2006.1661100.

[95] Takuma Otsuka et al. “Bayesian Unification of Sound Source Localization and Separation with Permutation Resolution”. In: vol. 3. Jan. 2012, pp. 2038–2045.

[96] Kai Wu et al. “Swarm Intelligence Based Particle Filter for Alternating Talker Localization and Tracking Using Microphone Arrays”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.6 (2017), pp. 1384–1397. DOI: 10.1109/TASLP.2017.2693566.

[97] Qi-bin Zhang, Peng Wang, and Zong-hai Chen. “An improved particle filter for mobile robot localization based on particle swarm optimization”. In: *Expert Systems with Applications* 135 (2019), pp. 181–193. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.06.006.

[98] Havangi Ramazan, Mohammad Nekoui, and Mohammad Teshnehlab. “A Multi Swarm Particle Filter for Mobile Robot Localization”. In: *International Journal of Computer Science Issues* 7 (May 2010).

[99] Ye Zhao et al. “Improved Rao-Blackwellised particle filter based on randomly weighted particle swarm optimization”. In: *Computers & Electrical Engineering* 71 (2018), pp. 477–484. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2018.07.055.

[100] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[101] Joonas Nikunen and Tuomas Virtanen. “Time-difference of arrival model for spherical microphone arrays and application to direction of arrival estimation”. In: *2017 25th European Signal Processing Conference (EUSIPCO)*. 2017, pp. 1255–1259. DOI: 10.23919/EUSIPCO.2017.8081409.

[102] Carlos De Marziani et al. “Simultaneous Round-Trip Time-of-Flight Measurements With Encoded Acoustic Signals”. In: *IEEE Sensors Journal* 12.10 (2012), pp. 2931–2940. DOI: 10.1109/JSEN.2012.2205675.

[103] Luke Calkins et al. “Distance Estimation Using Self-Induced Noise of an Aerial Vehicle”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2807–2813. DOI: 10.1109/LRA.2021.3060664.

[104] Kenneth Batstone, Magnus Oskarsson, and Kalle Åström. “Robust time-of-arrival self calibration and indoor localization using Wi-Fi round-trip time measurements”. In: *2016 IEEE International Conference on Communications Workshops (ICC)*. 2016, pp. 26–31. DOI: 10.1109/ICCW.2016.7503759.

[105] C. de Marziani et al. “Performance Analysis of an Acoustic Relative Positioning System”. In: *2007 IEEE International Symposium on Intelligent Signal Processing*. 2007, pp. 1–6. DOI: 10.1109/WISP.2007.4447647.

[106] Amjad Yousef Majid et al. “Lightweight Audio Source Localization for Swarm Robots”. In: *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. 2021, pp. 1–6. DOI: 10.1109/CCNC49032.2021.9369572.

[107] Bruce A. Garetz. “Angular Doppler effect”. In: *J. Opt. Soc. Am.* 71.5 (May 1981), pp. 609–611. DOI: 10.1364/JOSA.71.000609.

[108] F. Gustafsson et al. “Particle filters for positioning, navigation, and tracking”. In: *IEEE Transactions on Signal Processing* 50.2 (2002), pp. 425–437. DOI: 10.1109/78.978396.

[109] Jos Elfring, Elena Torta, and René van de Molengraft. “Particle filters: A hands-on tutorial”. en. In: *Sensors (Basel)* 21.2 (Jan. 2021), p. 438.

[110] *ReSpeaker 6-Mic Circular Array Kit for Raspberry P*. 2023. URL: https://wiki.seeedstudio.com/ReSpeaker_6-Mic_Circular_Array_kit_for_Raspberry_Pi/ (visited on 04/18/2024).

[111] *Grove - Speaker Plus*. 2022. URL: https://wiki.seeedstudio.com/Grove-Speaker-Plus/ (visited on 04/18/2024).

[112] C.E. Shannon. “Communication in the Presence of Noise”. In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21. DOI: 10.1109/JRPROC.1949.232969.

[113] James F. Kaiser. “Nonrecursive Digital Filter Design Using the I_0-Sinh Window Function”. In: *Proceedings of the 1974 IEEE International Symposium on Circuits and Systems*. Apr. 1974, pp. 20–23.

[114] K.M.M. Prabhu. *Window Functions and Their Applications in Signal Processing*. 1st. CRC Press, 2014. DOI: 10.1201/9781315216386.

[115] L. Kleinrock and F. Tobagi. “Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics”. In: *IEEE Transactions on Communications* 23.12 (1975), pp. 1400–1416. DOI: 10.1109/TCOM.1975.1092768.

[116] Peter Hart, Nils Nilsson, and Bertram Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/tssc.1968.300136. URL: https://doi.org/10.1109/tssc.1968.300136.

[117] Brad Rakerd et al. “Assessing the acoustic characteristics of rooms: A tutorial with examples”. en. In: *Perspect. ASHA Spec. Interest Groups* 3.19 (Jan. 2018), pp. 8–24.

[118] Wen Liu et al. “Fusion algorithm of improved fingerprinting/PDR/Map based on Extended Kalman Filter (EKF)/Particle Filter(PF)”. In: *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*. 2018, pp. 1–8. DOI: 10.1109/UPINLBS.2018.8559712.

[119] Md. Sabbir Rahman Sakib et al. “Improving Wi-Fi based indoor positioning using Particle Filter based on signal strength”. In: *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. 2014, pp. 1–6. DOI: 10.1109/ISSNIP.2014.6827597.

[120] Zhen-dong Liu et al. “An improved particle swarm optimization particle filter algorithm based on harmony search”. In: *2020 39th Chinese Control Conference (CCC)*. 2020, pp. 1661–1666. DOI: 10.23919/CCC50068.2020.9188964.

[121] Zhao Hui et al. “An Improved Particle Filter Based on UKF and Weight Optimization”. In: *2020 IEEE 3rd International Conference on Information Communication and Signal Processing (ICICSP)*. 2020, pp. 80–83. DOI: 10.1109/ICICSP50920.2020.9232021.

[122] Ananta Adhi Wardhana et al. “Mobile robot localization using modified particle filter”. In: *2013 3rd International Conference on Instrumentation Control and Automation (ICA)*. 2013, pp. 161–164. DOI: 10.1109/ICA.2013.6734064.

[123] Chao Cai et al. “Asynchronous Acoustic Localization and Tracking for Mobile Targets”. In: *IEEE Internet of Things Journal* 7.2 (2020), pp. 830–845. DOI: 10.1109/JIOT.2019.2945054.

[124] Patrick Lazik and Anthony Rowe. “Indoor pseudo-ranging of mobile devices using ultrasonic chirps”. In: SenSys ’12. Toronto, Ontario, Canada: Association for Computing Machinery, 2012, pp. 99–112. ISBN: 9781450311694. DOI: 10.1145/2426656.2426667.

[125] Kaikai Liu, Xinxin Liu, and Xiaolin Li. “Guoguo: Enabling Fine-Grained Smartphone Localization via Acoustic Anchors”. In: *IEEE Transactions on Mobile Computing* 15.5 (2016), pp. 1144–1156. DOI: 10.1109/TMC.2015.2451628.

[126] Shuai Cao et al. “Effective Audio Signal Arrival Time Detection Algorithm for Realization of Robust Acoustic Indoor Positioning”. In: *IEEE Transactions on Instrumentation and Measurement* 69.10 (2020), pp. 7341–7352. DOI: 10.1109/TIM.2020.2981985.

[127] Fabian Höflinger et al. “Acoustic Self-calibrating System for Indoor Smartphone Tracking (ASSIST)”. In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2012, pp. 1–9. DOI: 10.1109/IPIN.2012.6418877.