Master's thesis

2019

Therese Langseth

Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Marine Technology

Therese Langseth

Sonar-Based Monte Carlo Localization for Autonomous Underwater Vehicles

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Sonar-Based Monte Carlo Localization for Autonomous Underwater Vehicles

Therese Langseth

Marine Technology

Submission date:	June 2019
Supervisor:	Ingrid Schjølberg
Co-supervisor:	Stian Sandøy

Norwegian University of Science and Technology
Department of Marine Technology

NTNU Trondheim
Norwegian University of Science and Technology
Faculty of Engineering Science and Technology
Department of MarineTechnology

MASTER THESIS IN MARINE TECHNOLOGY

Spring 2019

FOR

Therese Langseth

Sonar-Based Monte Carlo Localization for Autonomous Underwater Vehicles

The increasing need for underwater inspection, maintenance and repair (IMR) poses challenges with regards to developing methods in order to do so in an efficient and safe way. The development of autonomous underwater vehicles will play an important role in the future for performing such tasks. One of the main challenges that comes with autonomy is situational awareness. The vehicle must be able to extract correct information from its set of on-board sensors to understand its surroundings. The ability to understand the surroundings of the vehicle is a basic requirement for the sense-plan-act methodology of robotics. An underwater vehicle must be able to know it's position relative to objects to-be manipulated or inspected, to perform its desired tasks and mission. To tackle these problems, this master thesis focuses on developing a robust localization algorithm for autonomous underwater vehicles with the use of sonar sensors.

Objective
The master thesis aims to investigate methods for localization using sonar sensor measurements. The outline of the work has been as follows:

1. Perform a literature review on relevant topics for this thesis. The main focus is probabilistic approaches to mobile robot localization and sonar sensors.
2. Investigate and analyse different methods for implementation of particle filters.
3. Verify the localization algorithm with data from both simulated and real-world data collected during experiments in the MC-lab at NTNU.
4. Discuss results and methodologies.
5. Draw the conclusions from the studies and discuss possible further research steps.

Supervisor	: Ingrid Schjølberg
Co-supervisor	: Stian Sandøy
Submitted	: January 15th 2019
Deadline	: June 11th 2019

Ingrid Schjølberg
Supervisor
[illegible]

Sammendrag

Hovedmålet med denne masteroppgaven er å undersøke bruk av akustiske sensorer for å lokalisere et undervannsfartøy. Bruk av bildesonar er ønskelig på grunn av sensorens lave kostnader og vekt. Masteroppgaven begynner med å gi en introduksjon til lokalisering av mobile roboter og utfordringene relatert til dette for et undervannsfartøy. En sonar-basert algoritme som bruker Monte-Carlo simulering har blitt utviklet for lokalisering av et undervannsfartøy. For å ta i bruk sonarmålingene i lokaliseringsalgoritmen så har det blitt gjort et nøye studie på bildesonaren og dens egenskaper.

Hovedbidraget til denne masteroppgaven er implementering og verifikasjon av et sonar-basert partikkelfilter. Algoritmen har blitt testet og verifisert på to forskjellige datasett. Det ene datasettet er et datasett fra Universitetet i Girona, og det andre ble samlet på NTNU sin Marine Cybernetics Lab (MCLab). Datasettet ble samlet ved bruk av undervannsfartøyet BlueRov2.

Resultater har vist at lokaliseringsalgoritmen klarer å lokalisere fartøyet gitt et a priori kart, uten noe kunnskap om fartøyets startposisjon. Selv med et kart som kun er en tilnærming av virkeligheten klarer algoritmen å estimere posisjonen med tilfredsstillende resultat. Ved å legge til et antall nye partikler når estimatet fra filteret er ansett som dårlig, klarer algoritmen å hente seg inn etter lokaliseringsfeil og konvergerer mot riktig posisjon. To metoder for å inkludere sonarmålinger i partikkelfilteret har blitt sammenlignet, ved å anvende de på datasettet fra MCLab. Kun en av metodene viste tilfredsstillende resultater.

Abstract

The overall objective of this thesis is to investigate the use of a mechanically scanned imaging sonar in localization for underwater vehicles. The imaging sonar stands out due to its low cost and weight and is therefore desirable in the field of underwater robotics. After an introduction about mobile robot localization, and its specific issues in the underwater domain, the thesis will focus on developing a sonar-based Monte Carlo localization algorithm for an underwater vehicle. In order to incorporate the sonar readings in the localization algorithm, a thorough review of the imaging sonar and the peculiarities of the sensor has been conducted.

The main contribution of this thesis is the implementation and verification of the sonar-based particle filter. The algorithm is tested and verified using two different datasets. The first dataset was a pre-existing dataset from the University of Girona, and the second dataset was gathered in the Marine Cybernetics Lab (MC Lab) at NTNU for the purpose of this thesis. The dataset in the MCLab was collected using the vehicle BlueRov2.

The results show that the localization algorithm is able to locate the vehicle given a *a priori* map, without any knowledge of the initial position of the vehicle. By adding a number of new particles when the estimate from the filter is considered as a poor estimate, the algorithm is also able to recover from localization failures and converge to the correct solution. Two different methods for incorporating the sonar measurements in the particle filter have been compared with the last dataset. Only one of the methods showed satisfactory results.

Preface

This thesis is the result of my work during the spring semester of 2019 at the Department of Marine Technology, Norwegian University of Science and Technology (NTNU). The work has been carried out under the supervision and guidance of Professor Ingrid Schjølberg.

I would like to thank my supervisor, Professor/Vice Dean Ingrid Schjølberg, for her guidance during this project and for motivating me throughout the whole period. I would also like to express my gratitude to my co-supervisor Stian Sandøy, for all his help and all the competence he has shared with me during this project.

Therese Langseth
Trondheim, June 7, 2019

Contents

List of Tables

List of Figures

Abbreviations

DGPS	=	Differential Global Positioning Systems
DVL	=	Doppler Velocity Log
EKF	=	Extended Kalman Filter
GPS	=	Global Positioning System
IMR	=	Inspection, Maintenance and Repair
MC Lab	=	Marine Cybernetics Lab
MRU	=	Motion Reference Unit
MSIS	=	Mechanically Scanned Imaging Sonar
SIR	=	Sampling Importance Resampling
SLAM	=	Simultaneously Localization And Mapping
SNR	=	Signal-to-noise ratio

Chapter 1

Introduction

The increasing need for underwater inspection, maintenance and repair (IMR) poses challenges with regards to developing methods in order to do so in an efficient and safe way. The development of autonomous underwater vehicles (AUVs) will play an important role in the future for performing such tasks. The objective by introducing autonomy in IMR missions is reducing the operational costs and the risk associated with humans, and meeting the increasing demand. Therefore, both the academic and commercial communities are focusing on developing robust and safe systems for AUVs.

The ability to understand the surroundings of the vehicle is a basic requirement for the sense-plan-act methodology in robotics. One of the main challenges for underwater robotics is self-localization due to the absence of GPS signals. To obtain an increased level of autonomy underwater, the vehicle must be able to extract the correct information from its set of onboard sensors in order to understand its surroundings. This means that the vehicle must be able to know its position relative to objects that should be manipulated or inspected, in order to perform its desired tasks, as well as navigating safely through the environment without damaging subsea installations. The complexity of the underwater environment poses challenges for the traditional sensors technology used for land and aerial vehicles. This is because water is a poor medium for propagation and penetration of electromagnetic frequencies, which restrains the use of remote sensing techniques. A different set of methods for sensing underwater must, therefore, be investigated.

The objective of this thesis is to develop a sonar-based self-localization algorithm for underwater vehicles, which is suitable for long missions and with a general *a-priori* knowledge of the environment. The proposed system will consist of a particle filter algorithm that is able to estimate the vehicles pose based on data from a sonar sensor and internal measurements. The proposed system will be validated based on data collected from the BlueRov2 in the Marine Cybernetics Lab, and a pre-existing dataset from the University of Girona.

1.1 Previous Work

Due to the importance of having accurate redundant position estimates in robotics, the field of robot localization has been widely investigated. Most of the research conducted within mobile robot localization is land-based and for aerial robots, which is based on a different set of onboard sensors, since ultra-sonic laser scans and cameras with high resolution can be used. Although the complexity of the underwater environment's constraints the use of the same sensors as for land-based vehicles, the same principals for localization is transferable to underwater robotics. The previous work for land-based localization will, therefore, be briefly mentioned in this section, as well as a more thorough investigation of previous work carried out in the field of underwater vehicle localization.

Traditionally, Kalman filters have proven to show promising results when dealing with sensor and pose uncertainties in robotics. However, the drawback with the Kalman filters is that it fails to recover from localization failures. In addition, the Kalman filter builds upon the assumption of a Gaussian distribution, which is often not valid, especially when working with underwater vehicles. Another approach to localization is the Markov localization, which is a variant of the Bayes-filter. Some of the early work within mobile robot localization for land-based vehicles was conducted by Burgard et al. (1996) and Dellaert et al. (1999). Burgard et al. (1996) introduced a grid-based Markov-localization algorithm which was able to localize the robot from noisy sensor readings and by using an occupancy grid representation of the environment. The drawback with this algorithm was the computational cost, which depended linearly on the grid size. Therefore, to tackle the disadvantages that lie in the nature of the Kalman filter and the Markov localization method, Dellaert et al. (1999) introduced the *Monte Carlo Localization*, which represents the probability density function by a set of particles which is randomly drawn from it. This makes it possible to obtain a localization method that can represent arbitrary distributions. The set of samples, which are called particles, are recursively updated by utilizing a technique called *Particle Filtering*. Dellaert et al. (1999) showed that the given method was able to determine the position of the mobile robot efficiently without knowledge of its initial location, and proved to reduce the memory required compared to grid-based Markov localization.

Several different techniques have also been proposed for AUV localization to overcome the lack of GPS signals under water. Caiti et al. (2005) addressed the problem by developing a localization algorithm based on receiving acoustic measurements from a set of floating buoys equipped with GPS measurements. The AUV was then able to locate itself by listening to acoustic pings, emitted by the buoys at regular time intervals, containing coded information about its GPS position. The limitation of this method is the need for enough surface buoys in the area of operation, and it is not evident for deep water missions since the signal will be lost at longer distances. Other acoustic-based localization techniques are the long baseline (LBL) and short baseline (SBL) systems. In both cases, the AUVs localization is determined by acoustic returns detected by a set of receivers. LBL systems require transponders to be installed on the seafloor to form an array, and a transceiver mounted on the vehicle. The AUV is then able to locate itself with respect to the transponders as explained by Collin et al. (2000). In SBS systems, a surface ship follows the AUV and is equipped with a high-frequency directional emitter which is able to accurately determine the AUV position with respect to the mother ship as done by Storkersen et al. (1998). The limitation of this solution is the need of a mother ship constantly floating in the surface above the AUV and the short range between the ship and the vehicle makes it not suitable for deep water missions around offshore structures. Further, particle filter techniques have also been utilized for AUV technology, although not as much as for land and aerial vehicles. Karlsson et al. (2003) proposed a particle filter based navigation system, suitable for both ships and underwater vehicles, by using a depth map and sonar sensors to support the INS system. The proposed system was able to successfully estimate the position of the vehicle given depth measurements and a terrain map. Further, Silver et al. (2004) presented a particle filter merged with scan matching techniques. This approach uses an approximation of the likelihood for sensor readings, based on the nearest neighbor distances, to approximate the probability distribution over possible poses in the particle filter.

1.1.1 Simultaneous Localization and Mapping

The problem of localization can be solved together with the mapping problem, which is known as Simultaneous Localization and Mapping (SLAM). Solving the full SLAM problem is out of the scope for this thesis, but a brief literature review in the field of SLAM has been conducted since it as an essential step in order to obtain increasing autonomy. This is because the SLAM algorithm can both provide more accurate position estimates and a map of the environment in case this is not available.

SLAM consists of filtering algorithms, which merges noisy sensor measurements with information from a kinematic or dynamic motion model of the system. The main objective for a SLAM algorithm is to estimate the position of the robot in a map, while recursively updating the map and re-using the information from the map to refine its position estimate. Filters commonly used for the SLAM problem can be divided into Gaussian filters, such as Kalman filters, or non-parametric filters, such as particle filters. The particle filter uses a finite number of possible positions for the robot to present the uncertainty distribution Thrun (2001).

One of the early stage SLAM algorithms were proposed by Smith et al. (1990), as a probabilistic framework based on a stochastic map approach using an Extended Kalman Filter. The limitation of this method is that the computational cost increases quadratically with the numbers of features Ribas et al. (2006). Since the Kalman Filter is based on the assumption of Gaussian noise in order to obtain optimal performance, different solutions to the SLAM problem has been proposed as an alternative for environments consisting of noise that can not be modeled as Gaussian noise. Thrun (2001) addressed the localization problem using Monte Carlo Localization, which is a particle filter that can accommodate arbitrary noise distributions.

1.2 Main Contributions

The thesis consist of two main contributions, which are

1. An implementation of a particle filter in MATLAB for localization and position tracking of an underwater vehicle using sonar data and measurements from the internal sensors on the vehicle.
2. An extensive verification of the algorithm using two different datasets collected by ROVs. The first dataset was gathered by the University of Girona and the second dataset is collected by the author in the Marine Cybernetics lab at NTNU.

1.3 Outline of Thesis

The outline of the thesis is as follows:

- Chapter 2: This chapter gives an introduction to the localization problem for mobile robots.
- Chapter 3: This chapter explains the principle of operations of the sonar sensor, and how to interpret sonar data.
- Chapter 4: This chapter explains the theory behind the particle filter and its relation to mobile robot localization.
- Chapter 5: In this chapter, the implementation of the particle filter is described, and two different measurement models are presented.
- Chapter 6: This chapter presents the experimental setup that was used in order to obtain two different datasets that will be used to verify the localization algorithm.
- Chapter 7: This chapter presents the results from the two experiments, and compares the results from the particle filter to the dead-reckoning estimates.
- Chapter 8: Discussion of the results.
- Chapter 9: Conclusion based on the results, and the authors' thoughts about future work within the topic of the thesis.

1.4 Contextualization

The 4th industrial revolution is already in play, driving industries towards digitalization and autonomy. Moving towards autonomy implies crew reductions, especially for removing humans from hazardous situations. An example of a hazardous situation is diving, were using automated subsea vehicles are a good replacement in terms of both safety and performance. Subsea vehicles can be used on oil and gas fields for pipeline inspections and subsea exploration, but also in growing industries like offshore wind energy. As the demand for subsea IMR increases, the operations and tasks are getting more complicated. This includes the demand for expanded ranges that the operation must cover, implying that vehicle localization is a requirement.

Chapter 2

Mobile Robot Localization

Mobile robot localization is the problem of determining the pose of a robot from sensor data relative to a known map of the surroundings. Being able to determine the vehicles pose in the operating environment is one of the key research areas for autonomous mobile robots. In an ideal world, the robot would be able to perfectly sense its own environment with a set of onboard sensors. Unfortunately, the faulty and noisy measurements in the real world can mislead the robot and give false information. Herein lies the problem of mobile robot localization, which is the fact that the robot is not able to obtain noise-free sensor readings for measuring its pose. Therefore, the pose has to be estimated by integrating data over time to get a sufficient estimate, since a single measurement is not enough to determine the robots pose with enough certainty.

This chapter will describe the main classifications of localization problems, and give an introduction to some of the most used localization algorithms. The information given in this chapter will be based on Thrun et al. (2005).

2.1 Classification of Localization Problems

This section will describe the main classifications of localization problems, as well as a short description of how the environment can be modeled and the difference between having a localization algorithm that can access the control system of the robot or not. According to Thrun et al. (2005), the localization problem can be divided into three types of problems with an increasing degree of difficulty. The simplest localization problem is *Position Tracking*, where the initial robot pose is known. Position Tracking is a local problem since the uncertainty is only given for a region close to the robots true pose. The second localization problem is the *Global Localization* problem where the initial pose of the robot is unknown. This is a more complex problem than the position tracking problem, since the pose error has no bounds, and having one probability distribution is usually not enough. The last and most difficult localization problem to solve is the *Kidnapped robot problem*. The Kidnapped robot problem takes the global localization problem further by kidnapping the robot during operation. The robot will then think that it knows where it is but, in reality, it does not. By testing the robots ability to recover from being kidnapped, one can measure its capability of recovering from global localization failures and it is, therefore, an important step towards building a fully autonomous system.

2.1.1 Robots Environment

The way the environment is modeled will significantly affect the difficulty of the localization problem. Environments can be either static or dynamic. In a static environment, the robot is the only object that will change position over time, and all other objects remain at the same location. Static environments have some mathematical properties that make it efficient for probabilistic estimation and is therefore preferred.
Dynamic environments include objects other than the robot that changes position over time. Objects of particular interest are those that persist over time, and that impact more than one sensor reading. Those objects that only affect one sensor reading can be treated as noise, and those that are not able to be to measured are not of any interest. Localization in a dynamic environment is more complex and difficult than in a static environment, and there are mainly two methods to tackle this challenge. The first one is to model the dynamics of the object in the state vector, and the second is to filter the sensor data in order to eliminate the damaging effects of un-modeled dynamics.

2.1.2 Passive Versus Active Approaches

Whether or not the localization algorithm can access the control system of the robot, is another parameter that characterizes different localization problems. We distinguish between passive and active approaches. In passive localization, the localization module only observes the robot that is operating and has no influence on the motion control or the mission of the robot. This can, for example, be a robot that is moving randomly or performing other everyday tasks. The other method, which is active localization, controls the robot in order to minimize the position error of a path following algorithm and/or avoiding the robot to move into a hazardous environment. Active localization approaches usually give a better estimate of the localization of the robot, since it is able to eliminate the ambiguity caused by having two or more equal belief of its position. This is done by actively moving towards information that will determine its pose certainly. However, active localization is limited by the fact that it requires control of the motion of the robot which is carrying out other tasks than only localization.

2.2 Localization Strategies

The most common approaches to mobile localization will be discussed briefly in this section. First, Markov-localization and EKF-localization will be explained, which are both unimodal Gaussian techniques. Further, Monte-Carlo Localization will be explained, which is a non-parametric method and is therefore not bounded to Gaussian distributions.

It is also important to remember that localization requires that a map of the area is given. The reader is recommended to read "Probabilistic Robotics" by Thrun et al. (2005) for further details if interested.

Markov Localization is a common approach to the localization problem. This approach is a probabilistic approach, based on Bayes theorem, which maintains a probability distribution over the space of all possible hypothesis as to where in the world the robot might be. Figure 2.1 illustrates the basic idea of Markov Localization for a robot which is moving in a hallway. Each picture portrays where the robot is actually located and its own belief of where it is, $bel(x)$, based on sensor data. The observation model, $p(z|x)$, is the probability of observing a door at any given location of the hallway and is used to update the belief. The robots initial belief is uniform over all poses, and the probability mass becomes more focused around the correct pose as the robot moves to the right in the hallway and acquire more sensor information. Markov Localization can be used to address the position tracking problem, the global localization problem and the kidnapped robot problem for static environments.

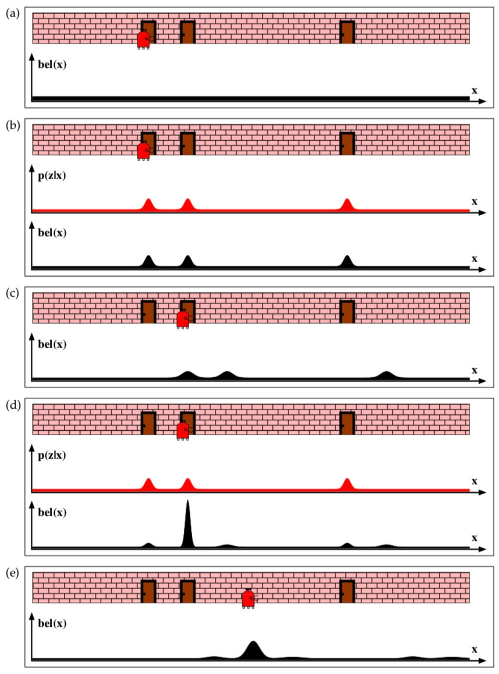


Figure 2.1: Illustration of Markov Localization from Thrun et al. (2005). Each picture frame presents the robots current belief $bel(x)$ as it moves forward. (a) and (d) additionally displays the measurement models $p(z|x)$, which is the probability of measuring a door at the different locations of the hallway.

Another approach to localization uses the Extended Kalman Filter (EKF) for feature-based maps. This is a special case of Markov localization, where the beliefs $bel(x)$ are represented by their first and second moment, the mean μ_t and the covariance $\sum_t$, which are updated for every new feature the robot observes. Another feature-based robot localization algorithm is the Unscented Kalman Filter (UKF). This approach uses the unscented transform to linearise the motion and measurement model, and the UKF can produce equal or better results than EKF for nonlinear systems. Both EKF and UKF are only applicable to the position tracking problem since a unimodal Gaussian assumption is not suitable for global localization problems.

Monte Carlo Localization (MCL) is yet another interesting approach which is applicable to both local and global localization problems. The MCL represents the belief state $bel(x)$ by a set of random particles covering the pose space and is therefore also called a particle filter. At first, all the particles are uniformly distributed in the pose space. The position of the particles is then updated by random samples from the motion model and represents all the different poses that the robot might have, depending on the sample size. Having more particles gives a higher certainty that the particle filter is able to cover all poses, and the number of particles to use is a trade-off for computational cost. After sampling the particles from the motion model, each particle will be assigned a weight as to how likely it is the true position compared with the observations that the robot make through its sensors. Since the particle filter is a non-parametric filter it can represent a probability distribution as a set of samples, and it is, therefore, able to approximate nearly any probability distributions. The MCL is also one of the simplest localization algorithms to implement, which makes it very popular.

Chapter 3

Sonar Sensor

Considering the limitations of usual sensors in the underwater domain, such as laser sensors and vision sensors, one needs to find a sensor that can overcome the peculiarities of the water medium. Given the excellent propagation of sound waves in water, an acoustic wave can travel a substantial distance without significant energy loss. This is utilized in a sonar sensor, which uses low-frequency sound waves in order to detect objects underwater. The focus of this thesis is to incorporate sonar measurements to use in localization for underwater vehicles. This section will, therefore, give an introduction to the operation principles of the sonar sensor by explaining the basics behind how the sonar sensor works and how to interpret the acoustic images received by the sensor. It is essential to understand these principals in order to process the sonar sensor measurements before using them in the localization algorithm.

Sonar systems are divided into active and passive systems, where the active sonar systems are used for object detection and localization of underwater vehicles, while passive sensors only receive sound waves emitted from the surroundings. Since passive sonar sensors are not applicable for localization, only active sonars will be described in detail. In an active sonar system, sound waves are propagating from a transmitter to a target and back to the receiver. The active sonar systems can also be divided into imaging and profiling sonars. A profiling sonar returns only the highest echo intensities, which provides an accurate cross-sectional profile of the surroundings. It is therefore mostly used in the mapping of the sea bed and in a structured environment such as along pipelines or bridge foundations. An imaging sonar sensor, on the other hand, returns all the echoes along the sonar beam, which gives a more detailed image of the robots surrounding and is therefore preferred for obtaining detailed information about the environment. This thesis uses a mechanically scanned imaging sonar, which will, therefore, be described further in the next section.

3.1 Mechanically Scanned Imaging Sonar

A mechanically scanned imaging sonar (MSIS) performs a scan in the 2D horizontal plane by rotating a transducer which emits fan-shaped beams at different orientations as seen in figure 3.1. The sector in which the sonar is scanning is normally configurable, ranging from a smaller sector to a full 360° scan, which makes it suitable for detecting objects in the environment surrounding an underwater vehicle. Each fan-shaped beam produces a scan with narrow horizontal beam width, for example, $1.8°$, and a wide vertical one which the figure 3.1 visualizes. When the emitted signal travels through the environment and collides with an object on its path, some of the energy in the mechanical sound wave is reflected back to the transducer. By using the time of flight for the returning wave, and knowing the speed of sound in water, the distance to the object is detected. By doing this, the MSIS can analyze the returning signal over a period of time in each direction to produce a series of echo amplitudes returning from a specific place along the transducer axis. From now, each of the amplitude measurements will be referred to as a *bin*, while the sets of bins obtained from a single emitted wave in one direction are referred to as a *beam*. The range of the sonar sensor is often configurable and is chosen to fit the designated operation.

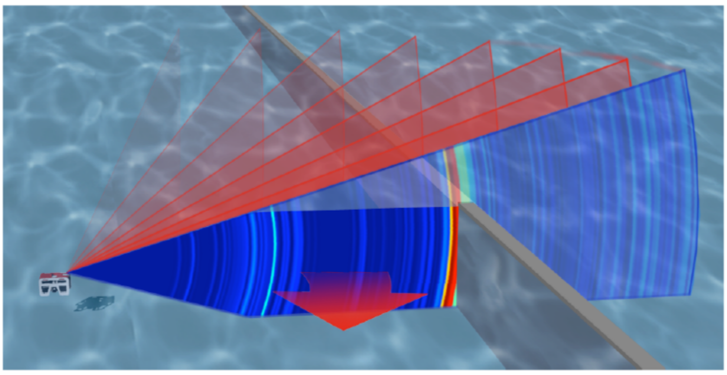
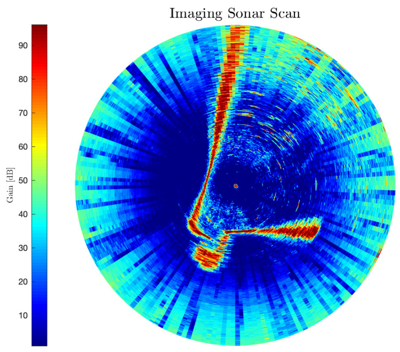
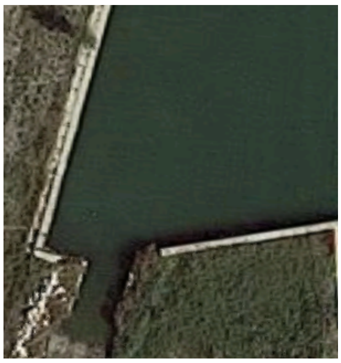


Figure 3.1: Imaging sonar Ribas et al. (2006)

Figure 3.2, shows a full 360° scan obtained by the Tritech Miniking Imaging from the dataset used in the first localization algorithm. The image has been produced by placing the sets of bins for each direction in a cartesian space and assigning different colors to the different intensities. The colors that fill the spaces between bins have been assigned through interpolation. Figure 3.2 is only used to simplify the interpretation of the sonar data and is not used in the particle filter directly.





a b

Figure 3.2: **(a)** Acoustic image obtained from real sonar data from the first dataset used in this thesis. The colorbar displays the different intensities of each beam, and one can clearly see the concrete walls which returns bins with high intensity and is therefore dark red. **(b)** Shows the flight photo of the marina where the acoustic scan is obtained. Flight photo taken from GoogleMaps.

3.2 Interpreting Sonar Data

In order to understand and extract the right information from an acoustic image, it is important to understand the process behind the generation of each beam. In some cases, the acoustic image can be almost identical with an optical image taken at the same place, and other times the acoustic image is perturbed with noise and other appearances that will be discussed in this section.

To get a deeper understanding of how the acoustic waves are reflected, the reader is recommended to look at figure 3.3 as a guideline for the following description. The process begins when the transducer emits a signal in a given direction. This signal then travels through the water volume without any impact from an object. Therefore, no noticeable echos are produced at first, only some noise from the robot itself which gives a peak at the beginning of each beam. Since the pulse is emitted in an arc-shape, it will quite soon reach the bottom of the environment if the vehicle is operating close to the ground. This will give the first significant returning echo, but due to the large incident angle between the signal and the surface, only a small amount of the mechanical energy is reflected and the measured intensity value is small. However, when the acoustic signal proceeds and hits an object, an increase in the measured intensity will appear. After the signal has been reflected by an object, an acoustic shadow will appear in the image. This is because the signal can not penetrate through the object, and one will discover a leak of echo intensity in the beam. The acoustic shadow produced by the object can be of particular interest since the length of the shadow can give information about object height Ribas et al. (2006). It is also important to emphasize that the large vertical beam width of the sonar only reproduces the object in a 2D dimensional plane, and the distance to an object might, therefore, be wrongly detected if it is reflected from a different altitude than the vehicle.

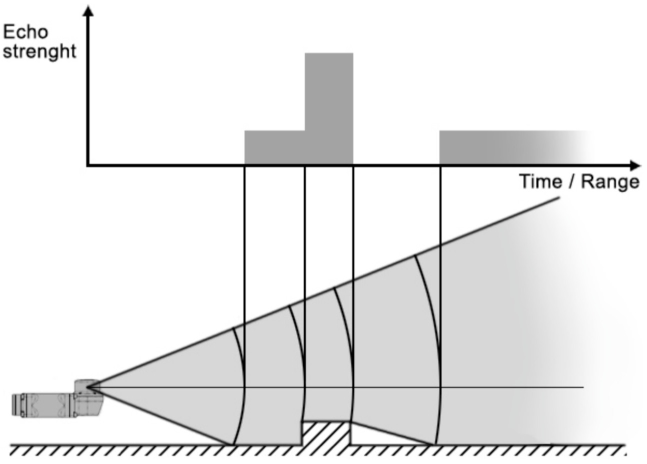


Figure 3.3: Visualization of the echo strength measured when the acoustic signal encounters objects (Ribas et al. (2006)).

Another interesting phenomenon occurs if walls or other large objects are present in the trajectory of the acoustic wave. Depending on the obstacle, this can potentially create phantoms and reflections that do not correspond to real objects, since a part of the acoustic wave is reflected in the opposite direction by the object. This effect usually takes place in confined spaces such as water tanks and test labs where the reflected waves do not disperse as easily as in outdoor environments. Figure 3.4 is an acoustic image from the Marine Cybernetics lab and portrays this phenomenon, where you can see the phantom fall at the far left side around $20[m]$ from the center. The two dark red straight lines close to the center of the image shows the real walls of the tank, which has a width of $6.5[m]$. Note that the other small intensity peaks which can be seen as the light blue areas are reflections from the ground of the tank.

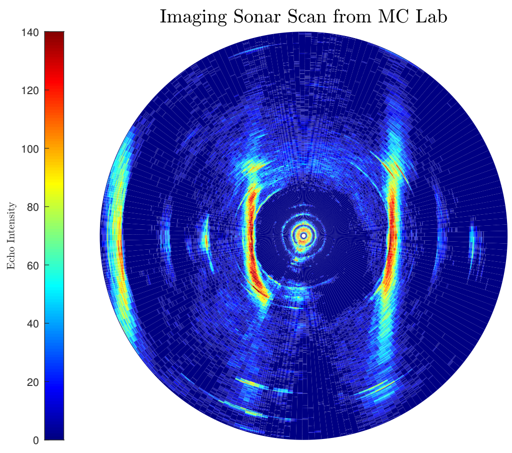


Figure 3.4: Sonar image from MC Lab.

3.3 Using Sonar Data for Localization

The raw sonar data must be processed in order to use it for localization. Measurements received by the sonar sensor are beside phantoms and reflections, also affected by noise. This means that a method for extracting the points corresponding to the real objects in the map must be developed, which will be described further in this section.

The noise influencing the sensor measurements can stem from the vehicle itself in the form of vibrations from the thrusters or as a result of currents generated into the transducer as stated in Ribas et al. (2006). The noise from the vehicle itself can be seen as a peak at the beginning of each beam. Figure 3.5 shows the echo intensities along one beam collected in the Marine Cybernetics lab with the Tritech Micron Sonar taken from the same scan as seen in 3.4. In this plot, one can clearly see the peak from the vehicle noise at around $0.5[m]$, the peak from the wall at around $2.5[m]$ and the phantom reflections at the end. This beam consists of 399 bins with a range of $10[m]$.

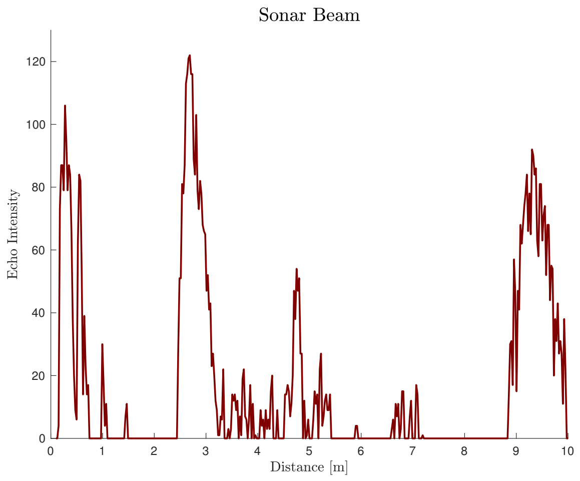


Figure 3.5: A sonar beam gathered with the Tritech Micron Sonar in the Marine Cybernetics Lab.

As seen from figure 3.5, where 3 peaks are clearly visible, the method for selecting the correct bin representing the actual object must be able to distinguish the object from the vehicle noise and other perturbations. The easiest method for finding which bin, b_i, belongs to an object is to select a threshold value, which means that a bin is set to belong to an object if it exceeds this value. The simplest form for threshold is

$$b_i = \begin{cases} 1, & for\ b_i \geq T \\ 0, & for\ b_i < T \end{cases} \tag{3.1}$$

where the value 1 belongs to an object being detected and is assigned to the bin, with index i, if the returning intensity is greater than the threshold value T. Using this method to select the bins is not robust against noise, as can be seen in figure 3.5, since the peak of the noise as an echo intensity close to the peak reflected from the wall.

More sophisticated methods for finding the threshold includes histogram shape methods, clustering based methods and gradient methods. The most common clustering based method is Otsus method, which assumes that the image contains two groups of pixels, foreground pixels and background pixels Bangare et al. (2015). The optimal threshold is then selected such that it minimizes the variance within the two classes. Histogram-based methods also assume that the image is divided into two classes, background and foreground, and *weighs* the histogram through an iterative process until the edges of the weighing scale meet Ramesh et al. (1995). When dealing with very noisy images, the weighing scale can be misplaced. This error can be minimized by removing the extreme values in the histogram. The last method, the gradient method, selects a threshold based on the value in the sample after the sample having the greatest gradient.

Chapter 4

Particle Filter for Localization

Of particular interest is the particle filter, as this is the chosen method to solve the localization problem in this thesis. Particle filters have become one of the most popular approaches in modern robotics, due to its simple implementation and its ability to work well across a broad range of localization problems. In particular, particle filters are able to represent complex multimodal beliefs, since it does not make any strong parametric assumptions of the posterior density, such as Gaussians. This makes the particle filter well suited to solve problems with global uncertainty, as well as hard data association problems. Data association problems arise when landmarks cannot be uniquely identified and is a key problem for range/bearing sensors and an important problem to consider. Although having these characteristics, the particle filter can be computationally expensive in order to obtain sufficient enough estimation. Fortunately, this can be adapted to each individual problem given the suspected complexity of the environment.

4.1 Basic Algorithm

A particle filter is a Bayes filter that represents a probability distribution $p(x)$ as a set of samples. The particle filter uses a prediction and an update cycle, just like the Kalman Filter, to estimate the state of a dynamical system from sensor measurements. The key idea of the particle filter is to represent the posterior, $bel(x_t)$, by a set of random samples drawn from it. Figure 4.1 shows how a non-parametric representation of Gaussian looks like. Instead of representing the distribution by a set of parameters, such as an exponential function that defines the normal distribution, the distribution is represented by a set of random samples drawn from it. This is an approximation of the distribution but makes it possible to represent any kind of distributions without being limited by the parameters defining it. Another advantage of the sample based distribution is that it is able to deal with non-linearities, as shown in figure 4.1. The figure shows how the particles are passed through a non-linear function $g(x)$, and the resulting samples are distributed according to the random variable Y.

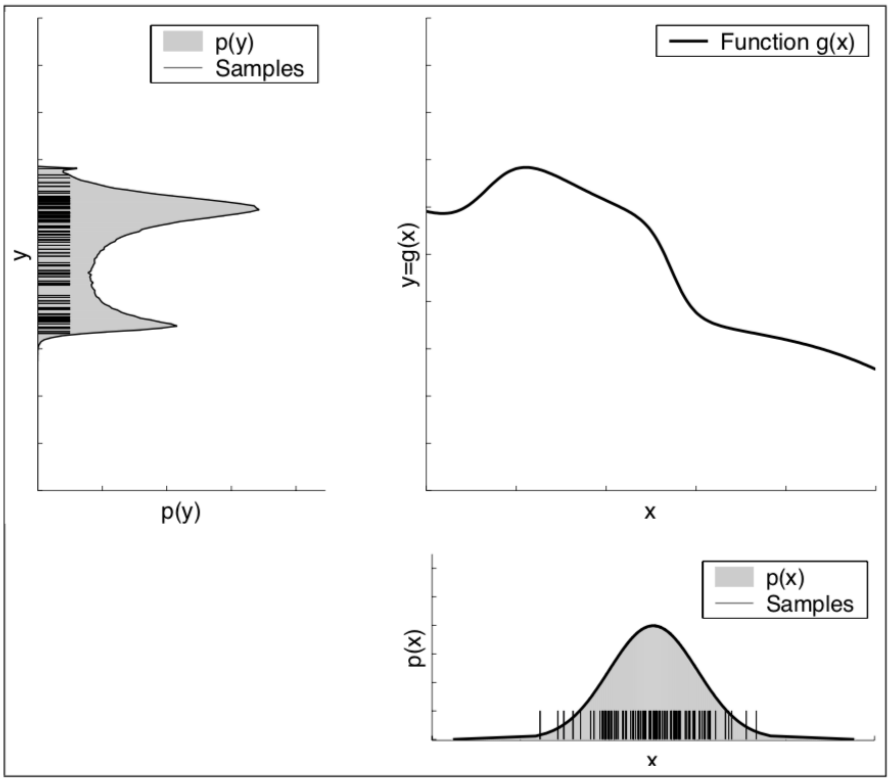


Figure 4.1: The particle representation used by the particle filter (Thrun et al. (2005)).

In a particle filter the posterior distribution is represented by a set of samples called *particles* and are denoted

$$\chi_t = x_t^{[1]}, x_t^{[2]}, ..., x_t^{[i]} \tag{4.1}$$

Each particle $x_t^{[i]}$ represents a possible pose that the robot might have at that instance. This means that each particle, $x_t^{[i]}$, is a hypothesis as to where in the world the robot might be at time t. N is the total number of particles used to represent all these possible states that the robot might have, and varies with the complexity of the localization problem. It is also possible to have the number of particles N, varying with the time t, in order to reduce the number of particles as the solution converges.

The particle filter estimates the posterior over all states in order to approximate the belief $bel(x_t)$ by the set of particles χ_t. The likelihood for a state hypothesis x_t to be included in the set of particles, χ_t, shall be proportional to its Bayes filter according to Thrun et al. (2005). This is given as

$$x_t^{[i]} \sim p(x_t | z_{1:t}, u_{1:t}) \tag{4.2}$$

It follows from equation 4.2 that if a region of the state space is highly populated by particles, it is more likely that the true state of the robot lies within this region. One should mention that the property of equation 4.2 is only valid if the number of particles goes to infinity. For a finite set of particles N, the particles are drawn from a slightly different distribution, but this is negligible if the amount of particles is big enough.

Just like all other Bayes filter algorithm, the current belief $bel(x_t)t$ is constructed recursively from the belief at the previous time step, $bel(x_{t-1})$. Since the beliefs are represented by a set of particles this means that the particle set at a time instant, χ_t, is constructed recursively from the previous set χ_{t-1}. By using the Bayes theorem and Markov assumptions, the posterior can be calculated according to Thrun et al. (2005) as

$$bel(x_{0:t}) = \eta w_t^{[i]} p(x_t | x_{t-1}, u_t) p(x_{0:t-1} | z_{0:t-1}, u_{0:t-1}) \tag{4.3}$$

where η is a constant and w_t is the particle importance weight. The importance weight is assigned to each particle according to its likelihood of being the correct pose estimate based on sensor information. This weight is then used further in a step called *resampling*, which will be explained in detail in section 4.3.

Algorithm 1: Basic particle filter algorithm

```
  Initialization:
1 $x_1^{[i]} \sim p_{x0}, i = 1, ..., N$                  // Generate initial particles
2 $w_1^{[i]} = 1/N$                                        // Set initial particle weights
3 for $t$=1,2,... do
4     for $i$=1,...,$N$ do
5         $x_t^{[i]} \sim p(x_t|u_t, x_{t-1}^{[i]})$       // sample from the motion model
6         $w_t^{[i]} = p(z_t|x_t^{[i]})$                   // update weights from measurement model
7     for $i$=1,...,$N$ do
8         $[x_t, w_t] = resample(x_t, w_t, N)$             // resample particles according to
          weights
```

To summarize the basics steps of the particle filter which can be seen in algorithm **Algorithm 1**, before going into more details of the specific parts, a short description of the main components is given in the list below.

- **Step 1 Prediction:** The next generation of particles is obtained by the particles from the previous iteration by sampling from the measurement model $x_t^{[i]} \sim p(x_t|u_t, x_{t-1}^{[i]})$
- **Step 2 Importance Weighting:** An individual importande weight is assigned to each particle according to the measurement model $w_t^i = p(z_t|x_t^{[i]})$. The weight must then be normalized in order to get the particles weight relative to other particles.
- **Step 3 Resampling:** Particles with low weights are replaced by samples with higher weights, in order explore more of the state space that is more likely to be the true position.

4.2 Particle Weighting

In order to determine if a particle is a good estimate of the robots real pose, it is assigned an importance weight. The importance weight is a way to incorporate the sensor measurements, z_t, in the particle filter since it is the likelihood of receiving the given measurements at each of the particle's position. By doing this the particle weights indicate how well the proposal distribution describes the target distribution. Assigning weights to the particles is an important step of the particle filter algorithm since the weighted set of particles represent the posterior distribution $bel(x_t)$. The particle weights are updated recursively for each measurement as

$$w_t^{[i]} = \frac{p(z_t|x_t^{[i]})p(x_t|x_{t-1}^{[i]})}{q(x_t|X_{t-1}^{[i]}, Y_t)} w_{t-1}^{[i]} \tag{4.4}$$

By choosing $q(x_t|X_{t-1}^{[i]}, Y_t) = p(x_t|x_{t-1}^{[i]})$ the updated particle weights are given as

$$w_t^{[i]} = p(z_t|x_t^{[i]})w_{t-1}^{[i]} \tag{4.5}$$

4.3 Resampling

The particle weighting that was described in the previous section will produce a particle set of independent simulation trajectories, $x_{1:t}^{[i]}$, with relative weights $w_t^{[i]}$. Without having a feedback mechanism to control the trajectories, they would diverge, as seen in figure 4.2, and all the relative weights will go towards zero except one that will go towards 1. Note that having a relative weight that goes to one is not an indicator of how close the particle is the real trajectory, it only says the particle is more likely than all the other particles. By introducing a resampling step, one can avoid experiencing sample depletion. The resampling is a probabilistic implementation of the Darwinian principle of *survival of the fittest* by reproducing the particles with higher weights and eliminating those with lower weights. This refocuses the particle sets to the region with a higher posterior probability, which makes it possible to explore more of the regions of the state space that are of particular interest and are more likely to be the real trajectory. The resampling used in this thesis is called the sampling importance resampling (SIR).

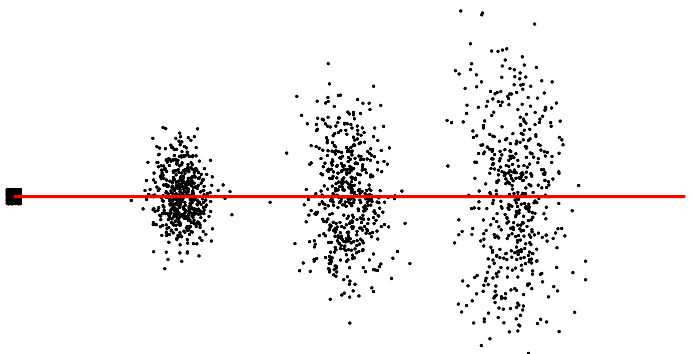


Figure 4.2: Particle evolution without resampling.

The standard SIR algorithm for particle resampling lets the particles with high weight to duplicate, while particles with low weight are less likely to survive. This can be explained by figure 4.3 if you imagine that the arrow will spin and stop at a random location. The particle at the location of where the arrow stops will be added to the new set χ_t. By spinning the arrow N times, we obtain the new set of particles, χ_t. It is easy to see from the figure that particles with low weight are less likely to make it to the new particle set, and particles with high weight will be chosen several times. In this way, only the feasible particles will survive for the next iteration, and we get a higher sample density in the areas of interest without changing the original PDF.

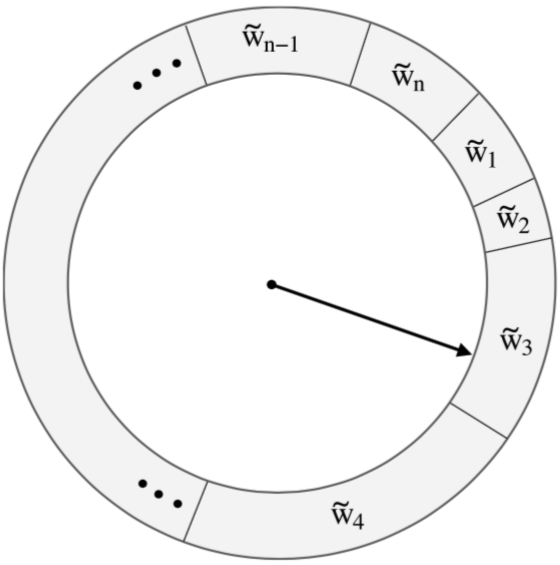


Figure 4.3: The principle of resampling based on particle weight (Øen Gustavsen (2018)).

4.3.1 Effective Resampling

When introducing the resampling step another problem occurs. The resampling step will destroy information and lead to increased uncertainty if performed at each time step. Effective resampling is a method to tackle this problem, which makes it possible to only resample when it is really needed.

From Gustafsson (2010), the effective number of samples can be approximated by

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t^{[i]})^2} \tag{4.6}$$

The value of this approximation lies between $1 \leq \hat{N}_{eff} \leq N$, where N is the number of particles. The upper bound $\hat{N}_{eff} = N$ is obtained when all particles have the same weight, which occurs when the particle filter is initialized and after the resampling step. The lower bound, $\hat{N}_{eff} = 1$, is attained when all the probability mass is devoted to one single particle. In order to use the effective number of samples as a condition for resampling in the particle filter, one must choose a threshold value N_t and execute resampling whenever $\hat{N}_{eff} \leq N_t$.

4.4 Motion Model

The motion model is a necessary component in order to implement the particle filter described in the previous section. Motion models incorporates the vehicle state transition probability $p(x_t|u_t, x_{t-1})$, which is essential for the prediction step in the particle filter. This gives the probability of a robots pose x_t given the previous pose x_{t-1} and the motion command u_t. The benefit of the particle filter is that it is sufficient enough to sample from the motion model distribution $p(x_t|u_t, x_{t-1})$, instead of calculating the posterior for arbitrary x_t, u_t and x_{t-1} (Thrun et al. (2005)). When sampling from a conditional density one is given u_t and x_{t-1} in order to generate a random x_t according to the motion model, instead of computing the probability $p(x_t|u_t, x_{t-1})$ for a given x_t generated through other means.

4.5 Measurement Model

The measurement model, which implements the probabilistic sensor model $p(z_t|x_t, m)$ in equation 4.5, is crucial for the correction step of the particle filter and necessary in order to correct the internal belief of the system. Without a correction step, the additive noise present in the motion model would make the estimation of the true state diverge. Therefore, it is necessary to design a robust and accurate sensor model in the particle filter. The sensor model defines the likelihood of a measurement z_t given the vehicle pose x_t at a time t with information of the environment from a map m. This section will discuss different methods for implementing the sensor model in the particle filter, with a focus on sonar sensors. It is important to emphasize that the sensor models described in will treat the sonar sensor as a range finders, which is visualized in figure 4.4. Instead of looking at the acoustic image, the range to an object is found by thresholding the sonar beams, which can be seen in the figure. In figure 4.4 this is done for one acoustic image obtained in the MC Lab.

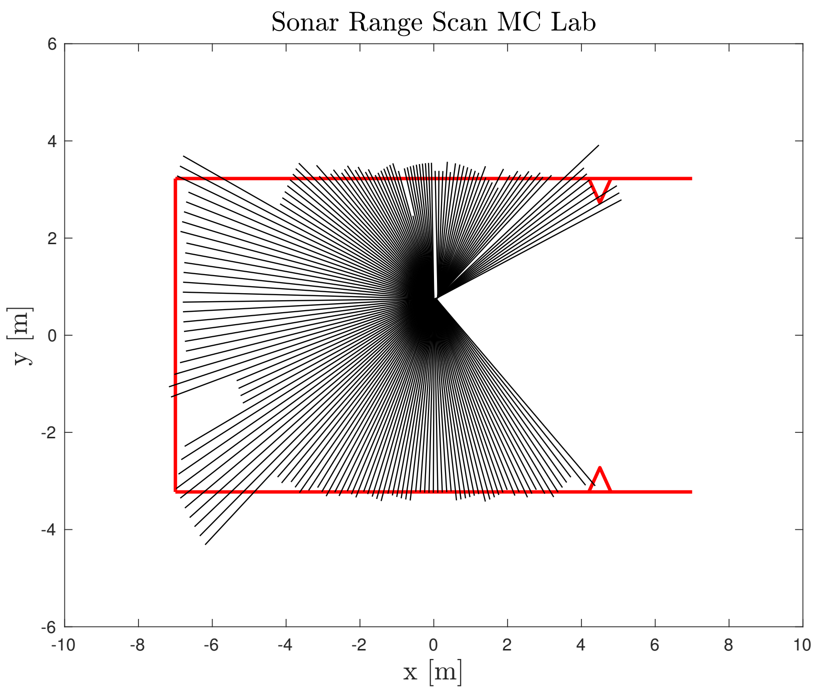


Figure 4.4: Sonar range scan made from an acoustic image by thresholding the sonar beams. The data used in this figure is obtained in the MC Lab.

4.5.1 Beam Models of Range Finders

The range finder model approximates the sonar beam as a one-dimensional ray and measures the distance to the closest obstacle. The noise in this model, meaning the error between the measured range z_t and the true range z_t^*, is often modeled as a narrow Gaussian with mean z_t and standard deviation σ_{hit}. This error arises from the limited resolution of the sonar range and environmental aspects. This distribution is given as

$$p(z_t|x_t, m) = \begin{cases} \eta\mathcal{N}(z_t, z_t^*, \sigma_{hit}^2) & \text{if } 0 \leq z_t < z_{max} \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

where η is a normalizing variable and the values measured by the sensor, is limited to the maximum range of the sensor z_{max}. The computation of the expected range, z_t^*, is normally done using a ray casting algorithm given the bearing of the measurement and the *a priori* map m. The ray casting algorithm will then return the first obstacle intersecting with the simulated ray in the given direction. This model works extremely well for laser sensor, which has a narrow beam, but becomes more limited while working with sonar sensor that has a wider horizontal beam width and also a vertical opening that will be completely discarded. Another characteristic to consider when using a range model for an imaging sonar sensor is that it will discard all the information from the returning intensities of the echos.

Thrun et al. (2005) described an approach to model the measurement likelihood for a range finder which is more detailed than the narrow Gaussian method explained previously in this section. All the equations is this section is from Thrun et al. (2005). This method is also based on performing ray casting in the map along the same axis as the emitted signal. Based on the distance to the nearest obstacle found by ray casting, z_t^*, a mixture of four different distributions captures the noise and error characteristic of the range sensor. The major component of this model is the same Gaussian distribution as in equation 4.7, which is centered around the expected range found from ray casting and with a variance σ_{hit}^2. Additionally, this model adds an exponential distribution in order to account for unexpected objects which are detected and not contained within the map. When treating the sonar sensor as a range finder, these readings would produce surprisingly short ranges compared to the map. Such measurements can, for example, occur from small dynamic objects in the vehicles environment or reflections from the bottom. The exponential distribution is decreasing with the measured distance from the sensor and is given by

$$p(z_t|x_t, m) = \begin{cases} \eta\lambda e^{-\lambda z_t} & \text{if } 0 \leq z_t \leq z_t^* \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

Furthermore, the model accounts for random measurement noise over the entire range of the sensor measurement given by a uniform distribution. For sonar sensors, this noise can often occur from phantom readings, especially in confined spaces with straight walls. The uniform distribution is constant over the whole range $[0; z_{max}]$, and is given by

$$p(z_t|x_t, m) = \begin{cases} \frac{1}{z_{max}} & \text{if } 0 \leq z_t < z_{max} \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

The last contribution to the beam model is account for sensor failures, which occur when detection of an object fails. For sonar sensor, this is likely to happen due to specular reflections of the signal at a large incident angle. A typical result of sensor failure is a max-range measurement, which therefore is modeled by a point-mass distribution centered around z_{max}, and can be expressed by

$$p_{max}(z_t|x_t, m) = I(z = z_{max}) = \begin{cases} 1 & \text{if } z_t = z_{max} \\ 0, & \text{otherwise} \end{cases} \tag{4.10}$$

The resulting measurement model described by Thrun et al. (2005) is then a mixture of these four different distributions, and a linear combination of the individual distributions are shown in figure 4.5. One can easily identify the four distributions in this figure, and the max-range point mass is modeled as a narrow uniform distribution.

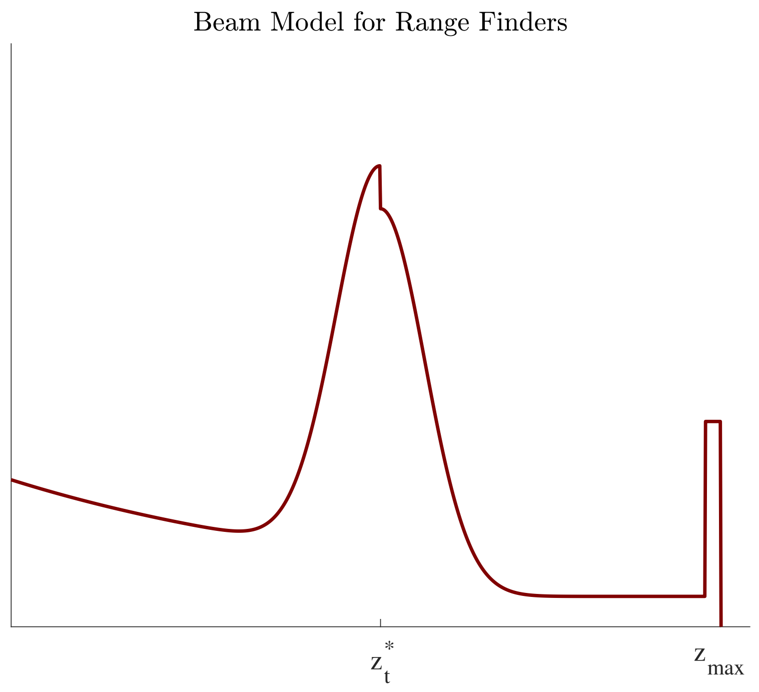


Figure 4.5: Pseudo-probability for range finders.

As discussed in Thrun et al. (2005), the beam model suffers from two major drawbacks. Even though the model is a proper physical implementation of the sensor characteristics, the *lack of smoothness* of the beam-based model poses challenges especially in cluttered environments. By lack of smoothness means that small changes in the pose x_t, can have an immense impact on the measurement model $p(z_t|x_t, m)$, which makes the model highly discontinuous in x_t. In environments with many small objects, this is effects is especially present, as small changes in the robots pose will impact the correct range of the sensor beam. Lack of smoothness leads to the fact that the state approximation can miss the correct state due to nearby states having a drastically different posterior likelihood. Further, the implementation of the beam model is computationally expensive as it requires a ray casting step for each particle when a new sensor measurement is received.

4.5.2 Likelihood-Field Model

An alternative method to the beam-based model for the implementation of the sensor model is called the likelihood-field model which is also presented in Thrun et al. (2005). This model overcomes a lot of the drawbacks that were discussed for the beam model in the previous section. On the other hand, this model lacks a true physical explanation and is a practical implementation of the sensor model, which gives a smoother posterior and is less computationally expensive than the beam-model.

The key idea is to project the endpoints of a given sensor scan into the global coordinate frame. In order to do this we need to know the particle pose at the time when the measurement is received, $x_t = \begin{bmatrix} x & y & \psi \end{bmatrix}^T$, the relative location of the sensor in the vehicle frame, $\begin{bmatrix} x_s & y_s \end{bmatrix}^T$, and the relative angular rotation of the sensor relative to the heading of the vehicle, θ_s. With a sensor measurement range, z_t, the global coordinate can be found as

$$\begin{bmatrix} x_{z_t} \\ y_{z_t} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} x_s \\ y_s \end{bmatrix} + z_t \begin{bmatrix} \cos(\psi + \theta_s) \\ \sin(\psi + \theta_s) \end{bmatrix} \tag{4.11}$$

The coordinates, $\begin{bmatrix} x_{z_t} & y_{z_t} \end{bmatrix}^T$, are only of interest when an object is detected by the sensor, and is used to find the Euclidean distance to the nearest obstacle in the map. Then the probability of a sensor measurement is given by a zero-centered Gaussian distribution with variance σ_{hit}, which captures the measurement noise (Thrun et al. (2005)). This can be written as,

$$p_{hit}(z_t|x_t, m) = \eta \mathcal{N}(dist; 0, \sigma_{hit}^2) \tag{4.12}$$

where dist is the Euclidean distance to the nearest object in the map from the endpoint found from the sensor measurement in equation 4.11. In addition to the zero-centered Gaussian distribution, uniform distribution can be added to model random noise in the model and a point-mass distribution for max-range readings in terms of failure.

The likelihood-field model has a smoother posterior distribution, since small changes in the robot pose will no effect the resulting distribution $p(z_t|x_t, m)$ as much as the beam model. This is because of the smoothness of the Euclidean distance compared to the ray-casting method. In addition to this, the likelihood-field model is more efficient as the most computationally expensive step, which is finding the distance to the nearest occupied cell in the map, can be pre-computed and stored for each cell within the map. For grid-maps, this accuracy will depend on the resolution of the map. Some of the drawbacks with this method is that it does not take into account unexpected short readings, which can be caused by dynamics objects in the environment or other sensor noise which is not accounted for. Further, the model can "see through walls", since the nearest neighbor cannot determine if the path to the given point is intercepted by another obstacle in the map.

Chapter 5

Implementation of the Particle Filter

This section will describe how the particle filter theory from chapter 4 is used in order to determine the pose of an AUV in a known *a priori* map with the available sonar measurements and measurements from DVL and IMU. The main focus of this thesis is the use of sonar data for AUV localization, and two different methods for implementing the sonar data in the measurement model of the particle have therefore been investigated. Further, the tuning of the particle filter will be described and the motion model.

5.1 Motion Model

The motion model used in the particle filter is a constant velocity kinematic model which predicts the particles state given the measurements, $p(x_t|x_{t-1}, u_{t-1})$. The kinematic model describes the vehicle motion, without considering the forces that caused it. This is a simplification since more complex models are hardly available. The kinematic models used in this implementation is also limited to three degrees of freedom, which is surge, sway and yaw. The measurements available on the ROV are linear velocities from the DVL and the rotational velocity from the IMU. The equations of motion are therefore given as

$$x_t = x_{t-1} + R(\psi_{t-1})\nu\Delta_t \tag{5.1}$$

where the vehicle state vector is $x = \begin{bmatrix} x & y & \psi \end{bmatrix}^T$ and the velocity vector is given by $\nu \begin{bmatrix} u & v & r \end{bmatrix}^T$. The velocities are considered constant within the time interval of Δ_t. Further, $R(\psi_{t-1})$ is coordinate frame transformation needed to derive the state transformation in the global reference frame, since the linear and angular velocities are expressed in the body-fixed coordinate frame. The transformation matrix is given by

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

From equation 5.1, the probabilistic model, $p(x_t|x_{t-1}, u_{t-1})$, can be derived. The linear and angular velocities are considered as the control input u_t, and the previous particle state is inserted for x_{t-1}. Since the system is subjected to process noise, random Gaussian white noise with zero mean is added to each of the velocity readings. The control signal is then given is

$$\hat{\nu} = \nu + \mathcal{N}(0, \sigma_\nu^2) \tag{5.3}$$

where $\sigma_\nu^2 = \begin{bmatrix} \sigma_u^2 & \sigma_v^2 & \sigma_r^2 \end{bmatrix}^T$. Each particle is therefore updated with a *random* velocity value, and integrated over the previous particle state x_{t-1}. This randomness helps spreading out the particles such that more of the state space is explored. The process noise is tuneable variables, and must be selected to obtain satisfactory results.

5.2 Measurement model

The measurement model will update the particle weights according to the probability that the pose of a particle is the true pose of the vehicle given the measurement that is received, $p(z|x_t^{[i]})$. The update of the particle weight is therefore given as the previous weight multiplied with this probability, which can be written as

$$w_t^{[i]} = w_{t-1} p(z|x_t^{[i]}) \tag{5.4}$$

Two different sensor models have been implemented based on the theory described in chapter 4. The first method is based on the beam model and the second method is based on the likelihood-field model. A description of these two methods is given below. Both methods treat the sonar sensor as a range finder and use this range in order to compute the likelihood. The range is found by selecting a threshold value for the sonar, and check if a portion of the values after the selected point is above a second threshold. This is to avoid that a noisy measurement can be treated as an object since real objects will provide a series of high-intensity bins.

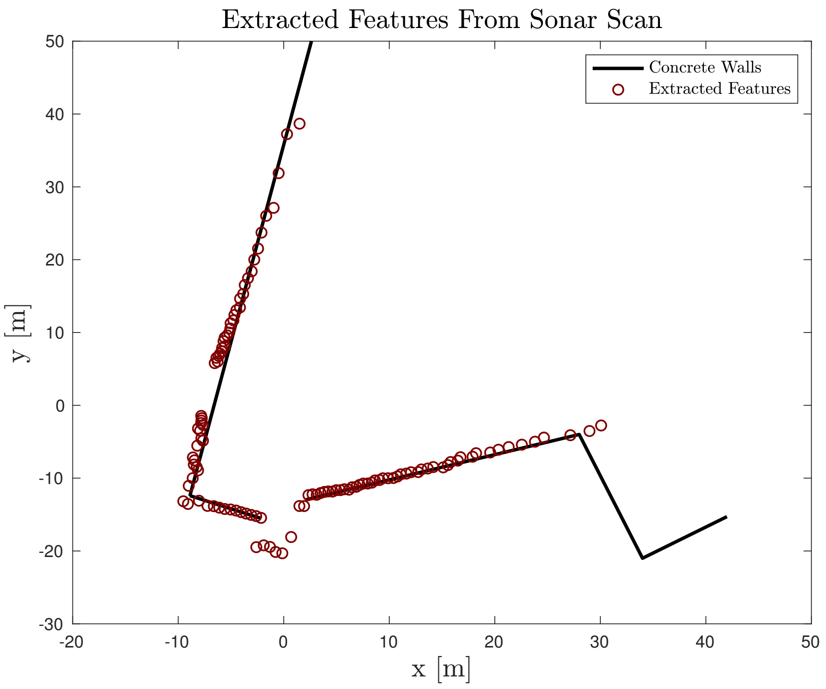


Figure 5.1: The red circles shows the features extracted after thresholding. The sonar measurements are taken from the first dataset and the black lines represents the concrete walls of the marina.

Figure 5.1 shows the performance of the chosen method for extracting the features in the acoustic image shown in figure 3.1. The chosen points align with the black lines that represent the concrete walls of the marina and is therefore a good method.

Method 1

The first measurement model is the beam-based model for range finders. This method is implemented according to the theory in section 4.5.1. If a sonar beam that is received detects an obstacle, the range to this object is found. Further, the expected range in the same direction as the sonar measurement is received from is found through a ray casting algorithm. In order to speed up the ray casting process, a lookup table is pre-computed for each grid cell and stored in a 3-dimensional matrix. This matrix contains the distance to the nearest object in any grid cell within the map for 200 angles. In order to look up the correct value, the sensors heading in the global frame is found by using the attitude of the vehicle and the correct index from the lookup-table is found by minimizing the difference between the pre-stored angles and the direction of the sensor measurement in the global frame. Figure 5.2 shows the ranges found from the look-up table plotted in the global frame together with the sonar measurements in the same directions. This figure also visualizes how the angle of incident affects the sonar measurements. When the acoustic signal has a large incident angle with the straight walls of the basin, the acoustic wave gets reflected in opposite direction, and the measured range detected by the sonar is larger than in reality which can be seen in the bottom left corner. In addition to this, the sonar sensor has a horizontal beam-width which is not accounted for. This means that the first reflections from the wall are actually reflected from a shorter distance than the one found from ray-casting for large incident angles, which can give shorter range readings.

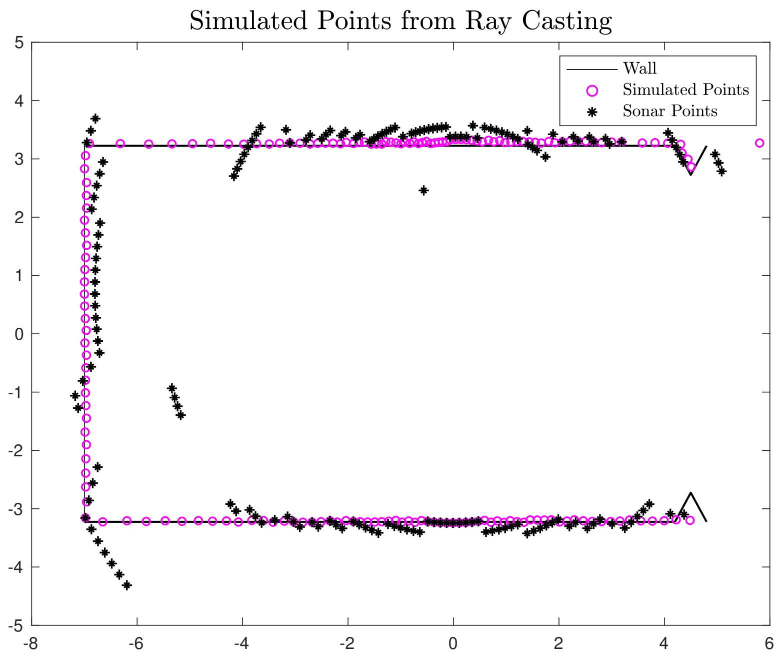


Figure 5.2: Expected data points from the ray casting algorithm compared to the real measured sonar data points found by extracting a distance value for each beam. The black lines represents the walls of the tank in dataset 2.

Method 2

The second measurement model is based on the likelihood-field model described in section 4.5.2. This model projects the measured sonar range from each proposed particle position into the global coordinate frame and finds the distance to the nearest object within the map. The probability of a wall distance is found by using a zero-mean Gaussian distribution. This gives the probability that a particle in a particular position is the true position, as a particle with a projected coordinate closer to the wall will get a higher probability than a particle with a projected coordinate far from the wall. Figure 5.3 visualizes this process, where 4 range measurements are obtained after thresholding the sonar beams. The sensor heading is given in the body-fixed frame, and the projection of the range readings is shown for the true position and three random particles. It is clear that the range readings plotted from the true vehicle position consist of the walls of the basin. From the random particles, it is clear that the distance from the range readings of particle 2 is closer to the wall than for particle 1 and particle 3. Particle 2 will therefore get a higher probability from the Gaussian probability distribution, and therefore a higher weight. The arrows pointing out from the particles and the true position are the body-fixed frames which show the orientation of the particles and the vehicle. Only 3 particles are displayed for simplicity.

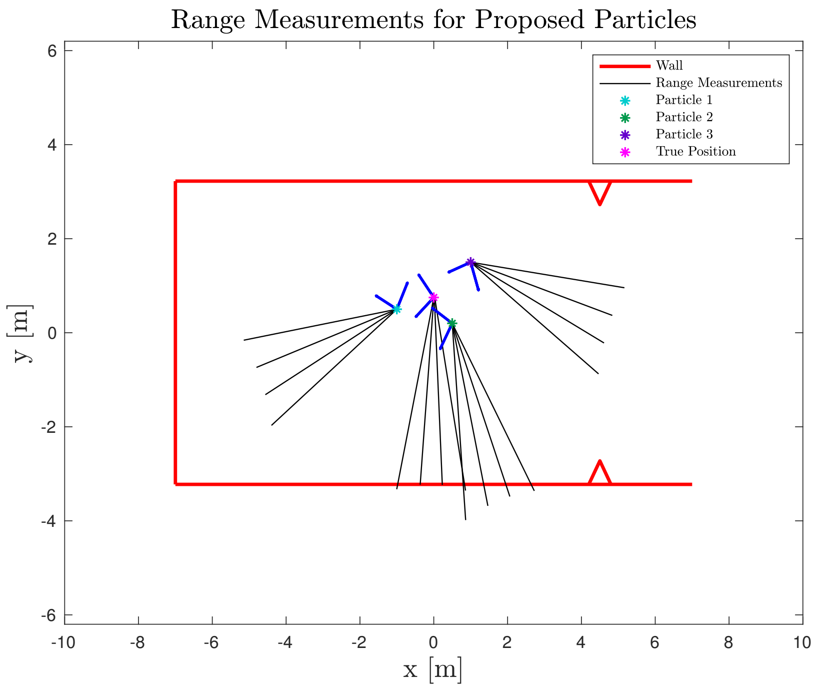


Figure 5.3: This figure displays the sensor method 2. The true vehicle state (pink) has received 4 range measurements, which is consistent with the wall of the basin. 3 random particles are presented in the figure, which has received the same range measurements. The body-fixed frames show the orientation of each particle. The shortest distance from the end-points of the range measurements to the wall determines each of the particle weights. Particle 2 (green) will therefore get a higher weight than the two other particles since the end-points of the beam is closer to the wall of the basin than for the two other particles.

The algorithm for finding the distance to the closest object is the map is the most computational expensive process of this model. To deal with this, a lookup table is constructed which contains the distance to the closest object for each grid-cell. This is done in MATLAB with the function *bwdist*. The accuracy of the lookup table depends on the resolution of the grid. A higher resolution gives a more accurate result, but it requires more memory. Figure 5.4, shows a graphical representation of the lookup-table for the map used in the first simulation for the data-set from the University of Girona.

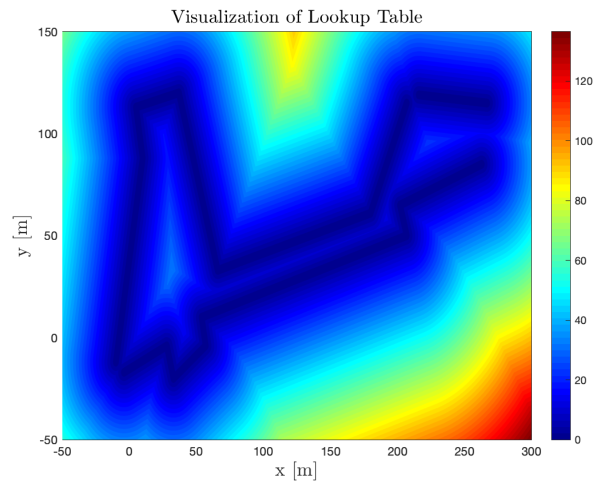


Figure 5.4: Grid map of the abandoned marina showing the closest distance to an object for each cell within the map.

Since a pure Gaussian distribution goes quickly towards zero, the particle weight would go towards zero due to outliers in the measurements. As seen in figure 5.2, some of the range measurements belong to points which are not reflected by the walls of the water basin and is most likely being reflected from the bottom of the tank or measurement noise. Such points would even give the particles that represent the true position a low weight since the weights are updated recursively. This is not desirable, and it is therefore useful to use a loss-function to avoid having a probability distribution that goes quickly towards zero. This is more practical consideration, rather than a theoretical one.
The loss functions are inserted in the normal distribution as

$$p(x) = \frac{1}{2\pi\sigma^2}\exp(-\rho) \tag{5.5}$$

where ρ is the loss function obtained from Sandøy (2019). Three different loss-functions were tested, which is given by

$$\rho_{cauchy} = \log(1+x) \tag{5.6}$$

$$\rho_{arctan} = \arctan(x) \tag{5.7}$$

$$\rho_{softOne} = 2 * (\sqrt{1+x} - 1) \tag{5.8}$$

$$\tag{5.9}$$

where $x = \frac{1}{2\sigma^2}dist^2$ and *dist* is the distance to the wall found from the lookup table which is visualized in figure 5.4. A mixture of loss-functions can also be used to achieve the wanted distribution. Using a combination of the Cauchy loss-function and the soft one loss-function gave good results. This distribution is shown in figure 5.5

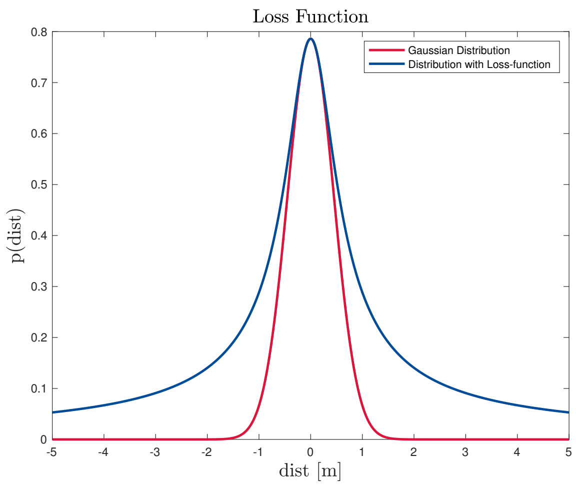


Figure 5.5: This figure shows how the combination of a Gaussian distribution with a mixture of loss-functions smooths out the resulting distribution. A mixture of the Cauchy loss function and the soft One loss fuction is used.

5.3 Resampling and Estimation

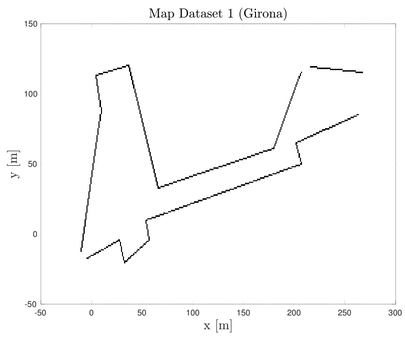
The resampling step used in the particle filter is the same as the general approach explained in section 4.3. This step is only performed when the effective number of particles N_t is lower than a selected threshold. Since the particle filter is not able to recover from the wrong convergence, an additional step is added to the resampling algorithm. If the maximum particle weight is lower than a selected threshold, a portion of the previous particles are not resampled and replaced with new particles drawn from a uniform distribution centered around the particle with the highest weight. For experiment 2, a new set of particles was instantiate if $w_{max} < 2.5\frac{1}{N}$. The new particles were given the same weight as the other particle from the resampling step, which is $w = \frac{1}{N}$.

The resulting pose estimation,$\hat{x} = \begin{bmatrix} x & y & \psi] \end{bmatrix}^T$, from the particle filter is given as

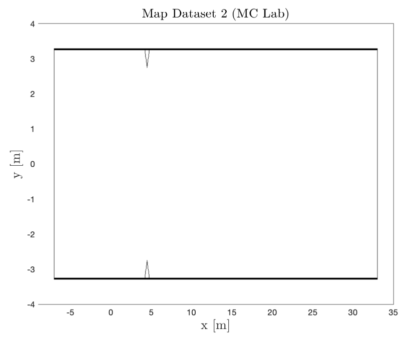
$$\hat{x}_{1:t} \approx \sum_{i=1}^{N} w_t^{[i]} x_{1:t}^{[i]} \tag{5.10}$$

5.4 Maps

In order to localize the position of the robot, an *a priori* map of the environment is needed as an input to the algorithm. This map is constructed as a matrix in MATLAB, where an occupied cell is assigned the value 1 and an empty cell is given the value 0. Figure 5.6, shows the map constructed for the first data-set from the University of Girona. This map is not an exact map of the environment, as this was not available. It is therefore constructed by using the point cloud from the sonar measurements mapped from the DGPS trajectory as a reference. Since the map covers a large area, the resolution was set to $r = 0.1[m]$ in order to save memory.



a



b

Figure 5.6: **(a)** Map of the abandoned marina used for localization in the first dataset. **(b)** Map of the Marine Cybernetics lab at NTNU.

5.5 Tuning of the Particle Filter

The tuning of the particle filter is rather intuitive and consist of tuning parameters for the process noise and the measurement noise. In addition to this, the number of particles in the filter must be chosen. This section will explain how the different tuneable variables impact the results of the particle filter.

The process noise determines how much the particles should be spread out between each iteration. If the measurements in the motion model are noisy and will give e poor estimate, this noise should be able to spread the particle cloud out in order to explore more of the state space. This is done by having a variance for the process noise, but the drawback is that it will not give a smooth estimation. On the other hand, if the motion model is very accurate, the noise can be reduced to obtain a more concentrated particle cloud and a smoother estimate. The two different sensor models have different parameters for tuning the measurement noise. Both measurement models are a mixture of probability density functions which have parameters that need to be selected. The most important variable to tune is the variance of the Gaussian distribution. Both the process noise and the measurement noise are selected based on trial and error.

Further, the number of particles needs to be chosen, which is a give and take between the accuracy of the filter and the computational effort needed to run the filter. A higher number of particles are able to cover a larger area of the state space, and thus yield a better estimate. On the other side, a larger number of particles leads to more calculations for each iteration and thus it requires more computational time. In the result section, a different number of particles have been tested and the performance is presented.

When tuning the particle filter, one pitfall might occur. If the particles are not spread out sufficiently between each iteration, the pose of all the particles will be very similar to each other. This is almost equivalent to having only one particle since they all provide the same information. If this occurs, the particles will be a poor estimation of the posterior distribution of the estimated state. In the worst case, the particle filter estimation can lose track and diverge from the true state. With the particles being really concentrated, it would not be able to recover after diverging, and the weight of the particles become irrelevant since the resampling step is only able to resample this one particle. In order to tackle this problem, an additional step is added to the resampling step when the maximum weight of the particles is lower than a selected threshold as mentioned in section 5.3. This threshold is also a tunable variable.

Chapter 6

Experimental Setup

The localization algorithm presented in the previous chapter has been verified using two different datasets. The first one is an extensive dataset from the University of Girona gathered in an abandoned Marina, and the second has been gathered in the Marine Cybernetics lab for the purpose of this thesis.

6.1 Dataset from Girona

In order to verify the performance of the localization technique described in Chapter 5, it has been tested using a data set obtained by the *Ictineu* AUV in an abandoned marina. The data set was gathered in March 2007 by the University of Girona and is useful to see if an algorithm is capable of utilizing the limited information provided by each sonar scan obtained in a large underwater environment. A flight photo of the marina where the data set was gathered is shown in 6.1. This section will give a description of the different sensors that the *Ictineu* AUV was equipped with.

Since all the different sensors are updated with a different frequency, interpolation is done between the measurements for the different sensors. An overview of the measurement frequencies is given in table **??**.

Sensor	**Frequency**
Tritech Miniking Sonar	15 Hz
DVL	1.5 Hz
MTi	10 Hz

Table 6.1: The update frequency of the different sensors on the Ictineu

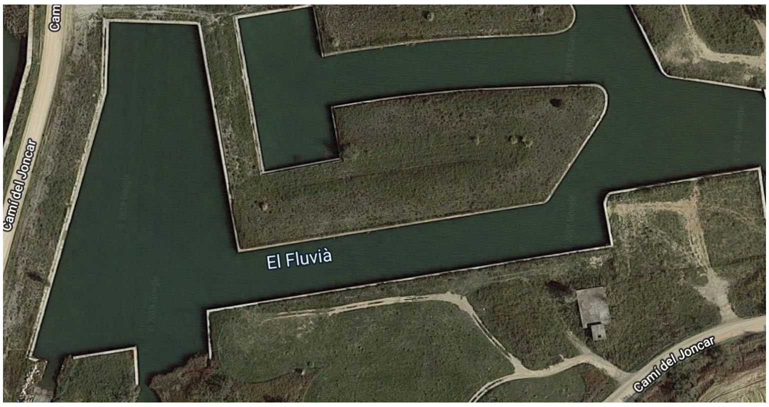


Figure 6.1: Flight photo of the abandoned marina retrieved from Google Maps.

Imaging Sonar

The imaging sonar that was used while gathering the dataset could rotate $360°$, as seen in 6.2, with a range of $50[m]$ and a resolution of $0.1[m]$. This results in each beam being represented by a vector containing 500 bins. As seen in 6.2, the angles of the imaging sonar is defined such that $0°$ points in negative x-direction, and must therefore be subtracted by $180°$ in order to get the angles in the desired vehicle frame. Each beam that the imaging sonar receives covers an angle of $1.8°$, which means that a full $360°$ scan is given by a set of 200 bins. With an update frequency of $15[Hz]$ it takes about $13.3[s]$ to obtain a full scan.

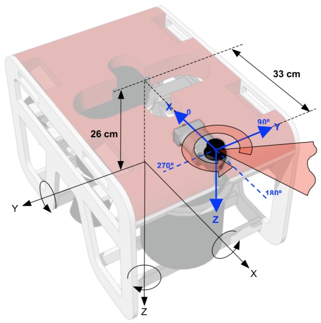


Figure 6.2: Sonar setup on the *Ictineau* vehicle.

Doppler Velocity Log

The AUV is equipped with a Doppler velocity log (DVL) which measures the vehicle speed over the ground with respect to the water in (x,y,z)-direction. The DVL operates at a frequency of $1.5[Hz]$ and has therefore a much lower update frequency than the imaging sonar. This means that the values from the DVL must be interpolated in order to account for the relative vehicle motion. This is implemented in the Extended Kalman filter. Moreover, the coordinate system of the DVL is not consistent with the vehicle frame and must be rotated. This is done by

$$\mathbf{R}^v_{DVL} = \begin{bmatrix} \cos\frac{-30\pi}{180} & -\sin\frac{-30\pi}{180} \\ \sin\frac{-30\pi}{180} & \cos\frac{-30\pi}{180} \end{bmatrix} \tag{6.1}$$

$$\mathbf{z}^v_{dvl} = \mathbf{R}^v_{DVL}\mathbf{z}^{dvl} \tag{6.2}$$

Motion Reference Unit

The AUV has a motion reference unit (MRU) which provides attitude and heading measurements, as well as rate of turn and acceleration measurements. In order to obtain drift free angular measurements, the motion reference measures the directions of gravity and the magnetic north. The MRU has an update rate of $10[Hz]$. Due to the high update frequency and the drift-free angular measurements, the MRU is the main source for the vehicles attitude estimation in the Extended Kalman filter.

DGPS

A DGPS was mounted in a buoy that was attached at the top of the vehicle during the experiment. This provides the data that is used for the ground truth in the experiment.

6.2 Experiment in Marine Cybernetics Lab NTNU

The author conducted experiments in the Marine Cybernetics lab (MC-lab) at NTNU in Trondheim in order to gather data to verify the proposed localization algorithm. Data was gathered by the remotely operated vehicle BlueROV2 in a pool. The pool is $40[m]$ long, $6.5[m]$ wide and $1.5[m]$ deep.

6.2.1 BlueROV2

The remotely operated vehicle BlueROV2 was used to gather sensor data in the MC-lab. BlueROV2 is a small working-class ROV with a 6-thruster vectored configuration, which makes it easy to maneuver in the desired degrees of freedom, which is mainly surge, sway and yaw for this operation. The BlueROV2 has a depth controller, which makes it stay at desired altitude during the vehicle operation. Communication with the ROV is done using a tether, which makes it possible to maneuver the ROV using a joystick, as seen in 6.4. The ROVs power supply is through an on-board battery-pack, which limits the duration of the operation.

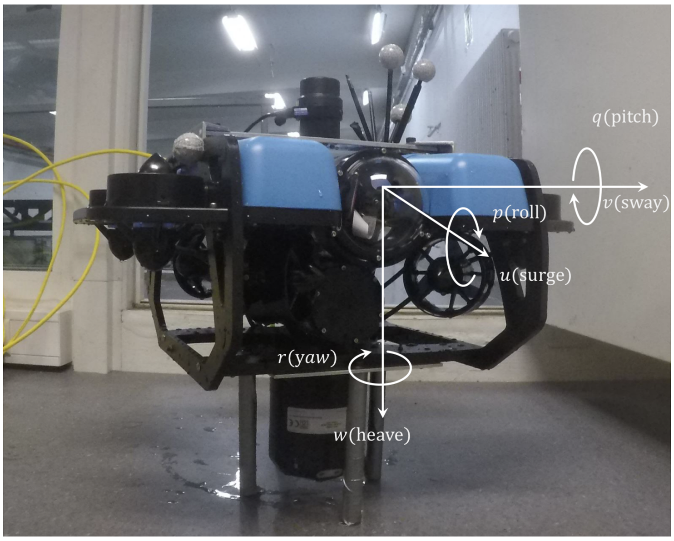


Figure 6.3: The BlueRov2 that was used to gather data in the Marine Cybernetics Lab. The Tritech sonar is the black cylindrical component that is mounted on top of the vehicle, and the DVL is the black cylinder at the bottom of the vehicle. Only the three degrees of freedom, surge, sway and yaw are considered in the localization algorithm.

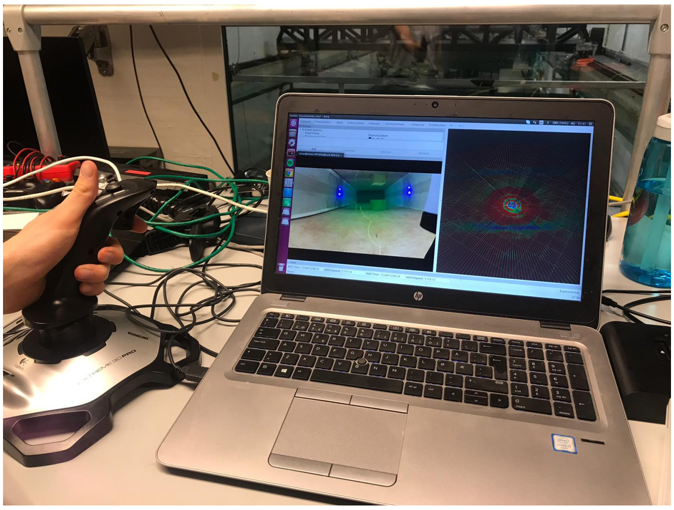


Figure 6.4: Maneuvering the BlueROV2 with Joystick and seeing the video stream from the camera on the vehicle as well as the sonar measurements.

6.2.2 Sensor Setup

The BlueROV2 was assembled with an advanced sensor configuration in order to get the data needed for the localization algorithm. The sensor setup consisted of DVL, Tritech Micron Sonar, camera and IMU.

The Tritech Micron Sonar is mounted at the top of the vehicle and is rotated 45° with respect to the vehicle frame, which must be accounted for. The sonar is configured with a range of $9.6[m]$, with a resolution of $0.2[m]$. The DVL is the black cylindrical shaped sensor at the bottom of the vehicle and is updated with a frequency of $5[Hz]$. Since all the measurements have a different update frequency, interpolated is done at each timestep in the particle filter. The measurement frequencies are given in table 6.2

Sensor	**Frequency**
Tritech Micron Sonar	24 Hz
DVL	5 Hz
IMU	50 Hz

Table 6.2: The measurement frequency of the different sensors on the BlueRov2.

6.2.3 Qualisys Motion Capture System

The pool in the MC-Lab is equipped with a Qualisys Motion Capture (QMC) system in order to measure the position and the attitude of the ROV during operation. Due to the high precision of the QMC, these measurements are used as a ground-truth to compare the results from the particle filter algorithm. The QMC system consists of 5 cameras attached to the wall of the pool, which can register special markers mounted on the vehicle. The markers are detected by reflecting infrared light which is emitted by the cameras. In order for the cameras to detect a rigid body, the markers are placed on the ROV in a special pattern which is stored as the body of interest in the Qualisys Track Manager (QTM) software. If the QMC system is not able to recognize the rigid body, the signal drops out, which is one of the main challenges with this system. Therefore, the markers are mounted on the ROV as clearly as possible, and parts of the ROV that could potentially reflect light are covered with masking tape. Due to the importance of the marker visibility, the operational space of the basin is reduced to a much smaller area when using QMC. During operation, it is important to stay within this area, and moving to close to either side of the basin makes the cameras loose track of the object.

Unfortunately, the Quliasys Motion Capture system could not be used when gathering the dataset, but two of the Qualisys cameras that are mounted on the wall was used in the *a priori* map. In figure 6.5 the BlueRov2 during operation can be seen. If the reader looks closely, it is also able to see one of the triangular shaped cameras mounted on the wall of the basin.

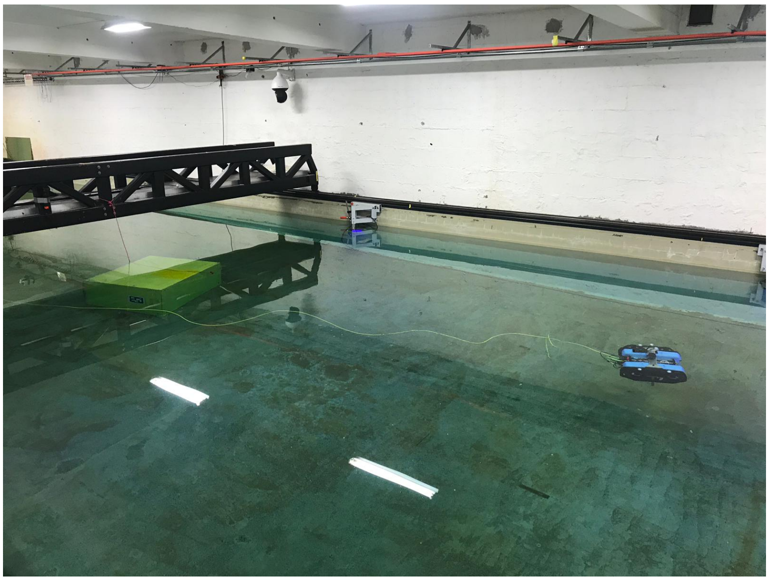


Figure 6.5: ROV during operation in the Marine Cybernetics Lab.

Chapter 7

Results

The implemented localization algorithm that was explained in chapter 5 has been verified by the extensive dataset collected by the University of Girona and data obtained from the Marine Cybernetics lab at NTNU. In this chapter, the results obtained will be represented. The dataset collected by the author at NTNU does not have a ground truth to compare the results but is still considered as a good contribution to the thesis.

7.1 Dataset 1 from University of Girona

Both the particles filter ability to perform position tracking and solving the global localization problem has been tested. In addition to this, the root mean square error for position tracking is given for a different number of particles.

The process noise used for this dataset is $\sigma_\nu^2 = \begin{bmatrix} 0.2 & 0.2 & 0.01 \end{bmatrix}^T$, for surge, sway and yaw respectively and measurement noise $\sigma_{hit} = 0.55$. The grid resolution was $r = 0.1[m]$ when constructing the lookup table, and only the likelihood-field based sensor model (measurement method 2) was used for this dataset.

7.1.1 Position Tracking

The particle filters ability to estimate the correct pose given an initial position is tested, which is called position tracking. The particles are spread out by a uniform distribution around the initial position with an uncertainty of $\pm 1[m]$ in x and y, and $\pm 5.7°$ for heading uncertainty. To investigate how the number of particles influences the position tracking abilities of the particle filter, several different numbers of particles have been tested. The result can be seen in 7.1, where the particle filter estimate is compared to the ground truth and the dead-reckoning estimate. The walls of the marina that are used in the *a priori* grid-map is plotted in order to get a more visual representation of the results. Figure 7.2 shows the absolute translational error in x- and y-direction for the position estimate compared with the dead-reckoning throughout the operation.

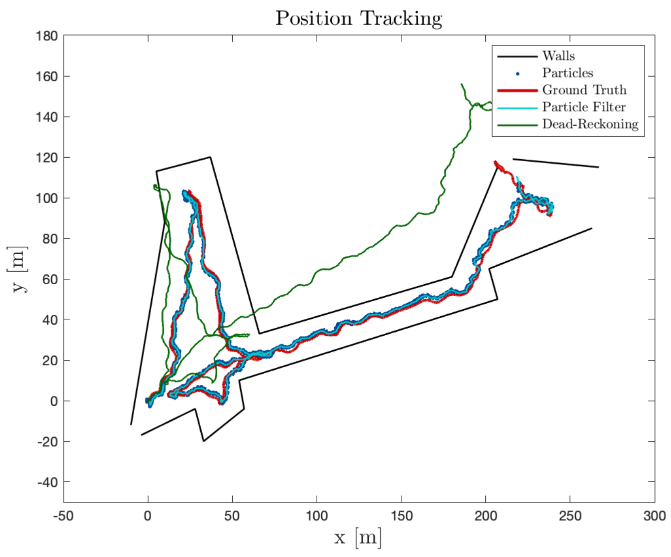


Figure 7.1: Position estimate from DVL and MTi (dead reckoning), compared to the DGPS ground truth and estimate from the particle filter localization algorithm. The particles are plotted for every 100 time-step (dark blue dots).

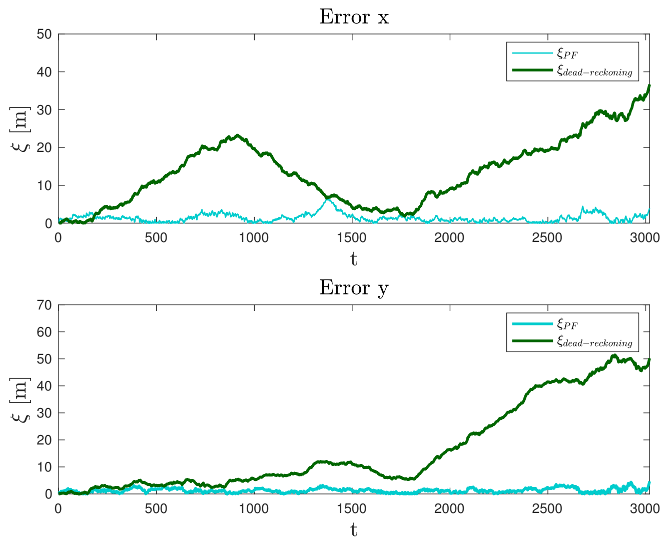


Figure 7.2: Absolute translational error in x- and y-direction for the particle filter estimate and the dead-reckoning estimate.

In order to see how the amount of particles used in the filter affects the accuracy of the position estimate, the average root mean square error for 10 runs with different sample size is found. The results is shown in figure 7.3. The error bars indicate a 95% confidence interval using the t-distribution.

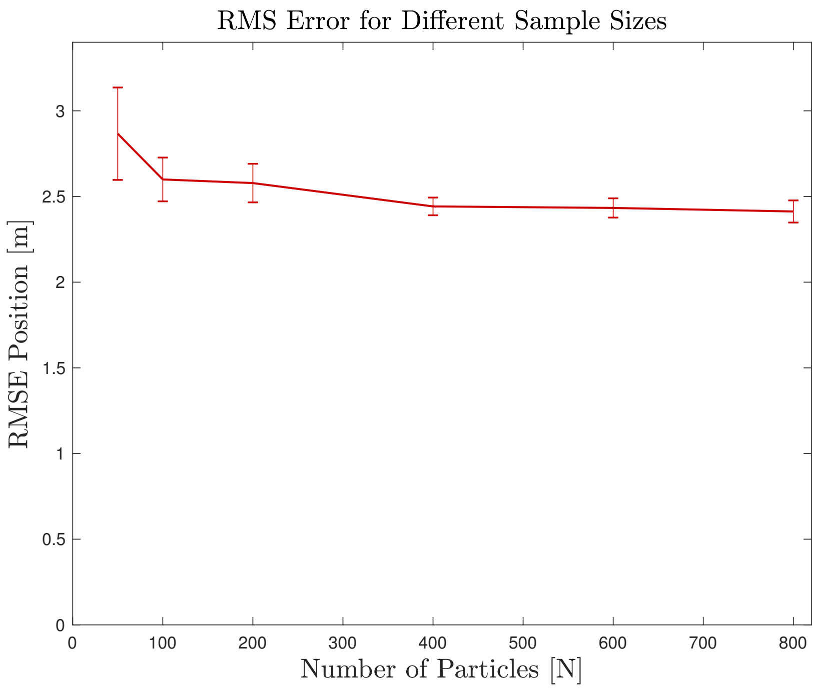


Figure 7.3: The figure shows the average root mean square error for different number of particles over 10 runs. The bars indicate a 95% confidence interval for the 10 runs.

7.1.2 Global Localization Problem

The particles filter ability to converge to the correct position without any knowledge of the initial pose of the vehicle is tested by spreading the initial particles uniformly over a large area. The initial particles are spread out over $6825[m^2]$, and the initial heading of the vehicle is unknown and drawn uniformly from $\psi \in [-\pi, \pi]$. The number of particles used is 1500. In addition to this, the particle filter algorithm was set to only resample 80% of the particles, and add 20% new particles drawn from a uniform distribution $\pm 5[m]$ in x, $\pm 5[m]$ in y and $\pm \pi$ in heading around the position of the particle with highest weight, when the maximum weight of the particles was below $w_{max} < 2.5\frac{1}{N}$.

The results from the global localization problem can be seen in figure 7.4, where the particles are dark blue dots in the figure and plotted for every 100th timestep. The estimate from the particle filter is given by the light blue line, and the ground-truth is plotted in dark red for validation. To see the outstanding performance of the particle filter, the dead-reckoning estimate is included in the plot (green line) for comparison.

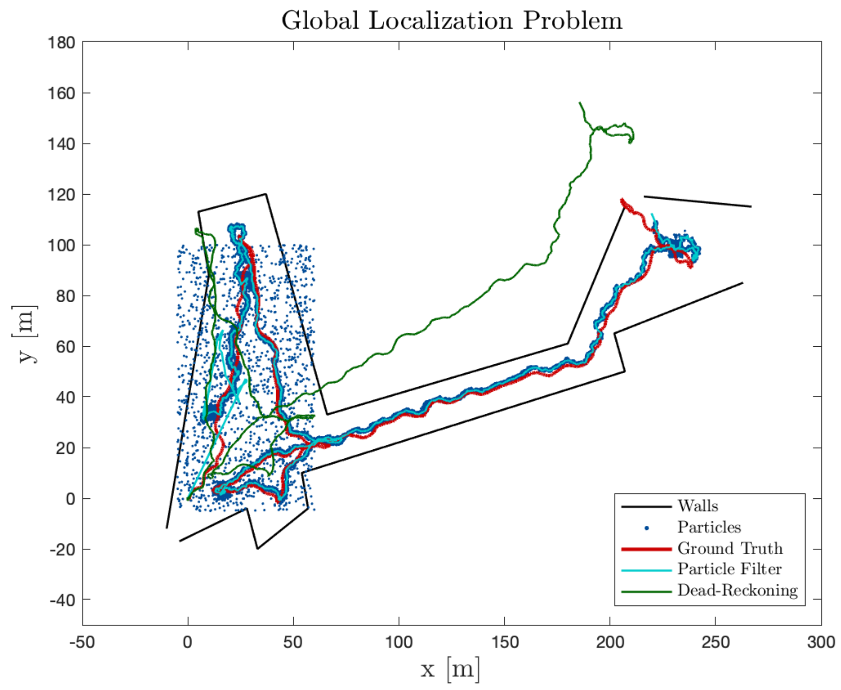


Figure 7.4: Position estimate from the particle filter for the global localization problem compared to the ground truth. The initial particles are spread out over $6825[m^2]$. The dead-reckoning estimate is plotted in green.

The translational error in x- and y-direction for both the particle filter estimate and the dead-reckoning estimate, from figure 7.4, is shown in figure 7.5. The red line from figure 7.4 is used as the ground truth for calculating the error.

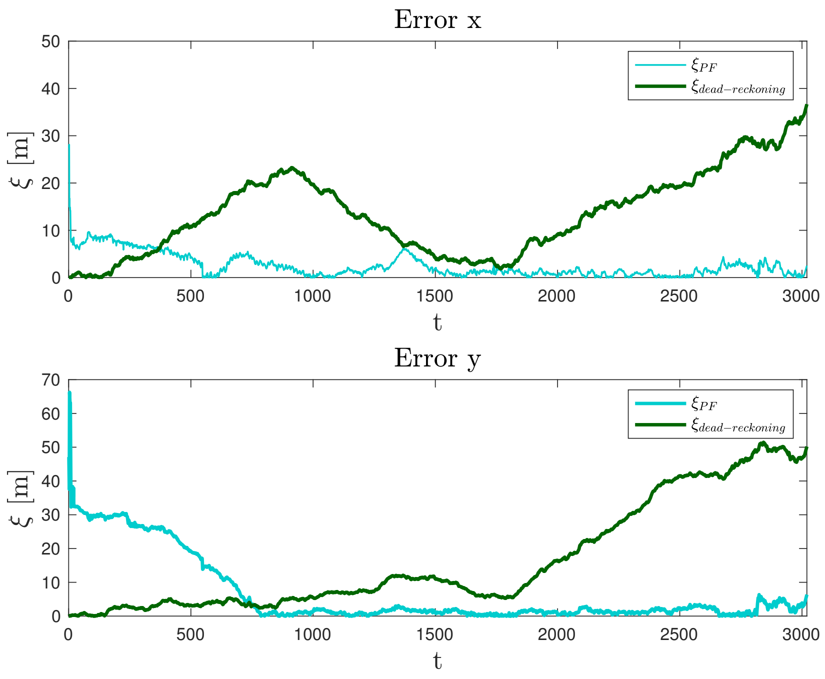


Figure 7.5: Position error for the global localization particle filter estimate

To investigate the convergence rate of the particle filter with a different number of particles, 20 runs were made for each amount of particles and the success rate of the algorithm is shown in table 7.1.

Particles (N)	Convergence	Percentage
1000	15	80 %
1500	17	85 %
2000	18	90 %
3000	20	100 %

Table 7.1: Convergence rate for the global localization problem for different number of particles over 20 runs.

7.2 Dataset 2 from MC Lab

The dataset from the MC lab does not have a ground-truth to compare the results, and a translational error can therefore not be found. It is therefore assumed that the odometry estimation, found from integrating the linear and rotational velocities, gives a good position estimation to compare with the particle filter estimate. In addition to this, the point cloud obtained from the position estimates is used to measure the performance, as the point cloud should be centered around the walls of the basin if the estimation is the correct position. Both of the measurement models described in section 5.2 has been tested on this dataset. First, the position tracking abilities is verified, and then particle filters capabilities of solving the global localization problem are tested.

7.2.1 Position Tracking

The position tracking abilities of the particle filter was tested as with the first dataset. The results for both the measurements methods are given below.

Measurement Method 1

The results from the measurement method 1 are presented here. Sine the method failed when increasing the uncertainty of the initial heading, only the position tracking capabilities of the method will be represented. The grid size used was $r = 0.05$ for making the lookup-table, and the process noise is $\sigma_\nu^2 = \begin{bmatrix} 0.04 & 0.02 & 0.008 \end{bmatrix}^T$. The particles are spread out by a uniform distribution around the initial position with an uncertainty of $\pm 0.5[m]$ in x and y, and $\pm 5°$ heading uncertainty. 600 particles are used. The resulting particle filter estimate can be seen as the light blue line in figure 7.6, and the particles are plotted as darker dots every 50th timestep.

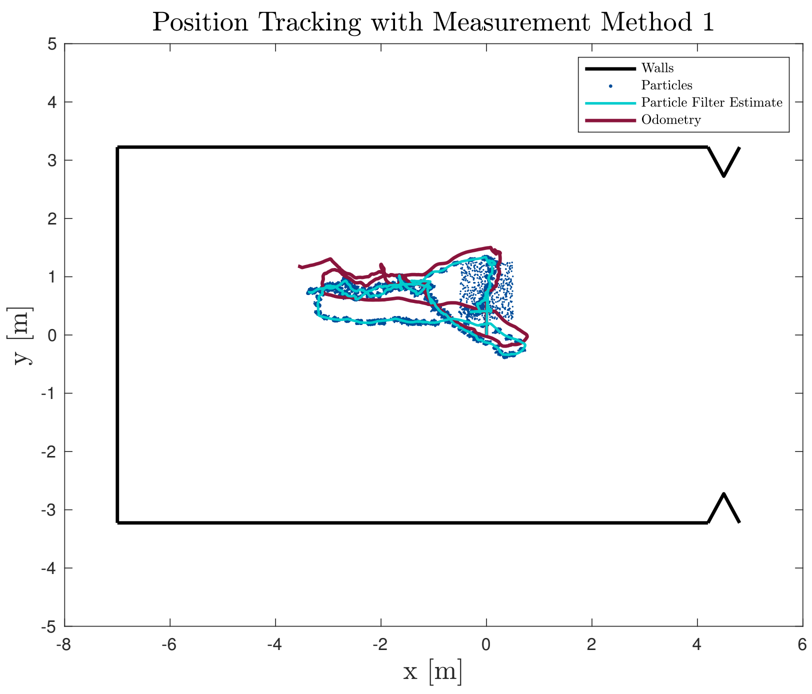


Figure 7.6: Position tracking for measurement method 1.

The resulting sonar point cloud from the estimation can be seen in figure 7.7, in light blue, compared to the point cloud from the odometry estimate in dark red. The average shortest distance for all of the points in the point cloud is $0.22[m]$, compared to $0.25[m]$ for the odometry point cloud.

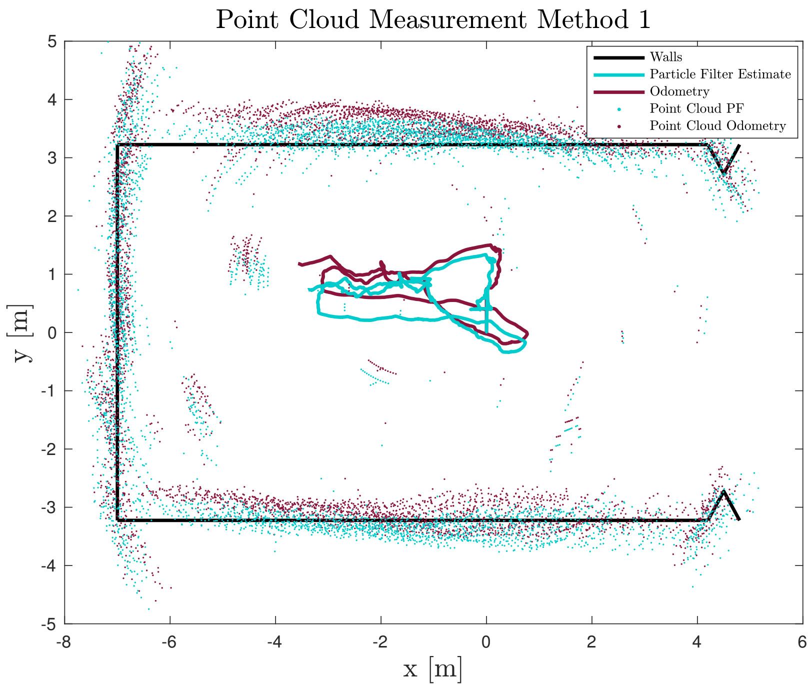


Figure 7.7: Sonar point cloud from position tracking with the measurement method 1.

Measurement Method 2

The position tracking abilities of the measurement model 2 are presented here. The grid size used is $r = 0.01$, and the process noise is $\sigma_\nu^2 = \begin{bmatrix}0.04 & 0.02 & 0.008\end{bmatrix}^T$ and measurement noise $\sigma_{hit} = 0.35$. The particles are spread out by a uniform distribution around the initial position with an uncertainty of $\pm 0.5[m]$ in x and y, and $\pm 5°$ heading uncertainty. 600 particles are used. The resulting particle filter estimation is seen in figure 7.8. In this figure, all the particles are plotted every 50th timestep as dark blue dots. In addition to this, the particle filter estimate is the lighter blue color on top of the particles and the odometry estimate is the dark red line. The walls of the water tank are included for visualization, and the triangular shapes are the Qualisys cameras which are mounted on the walls of the basin.

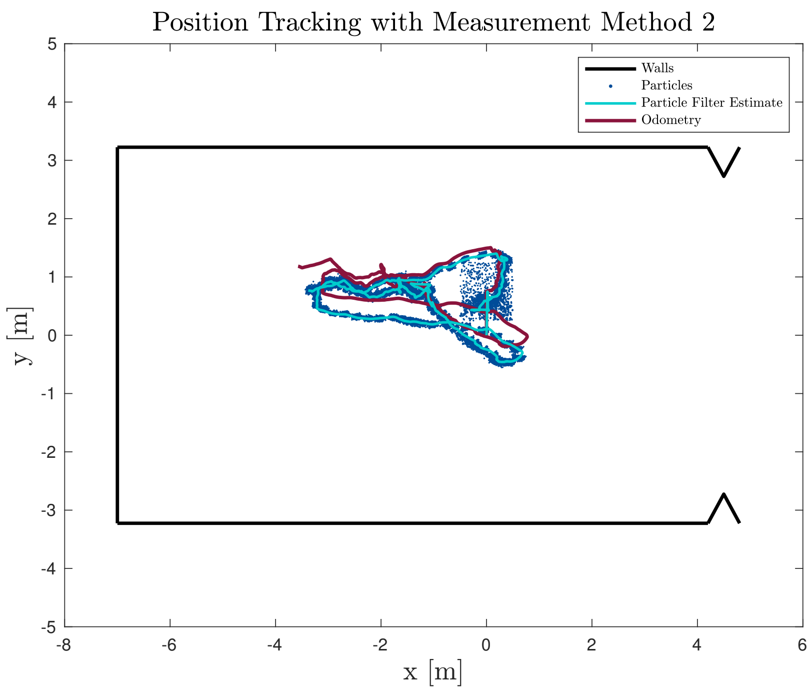


Figure 7.8: Position tracking for measurement method 2.

Since the dataset does not have a ground-truth, the resulting sonar point cloud from the filter estimation is used to evaluate the performance of the filter. This is seen in figure 7.9, where the blue point cloud is for the particle filter estimation, and the red point cloud is for the odometry estimation. The average shortest distance for all the points in the cloud to the wall is $0.24[m]$.

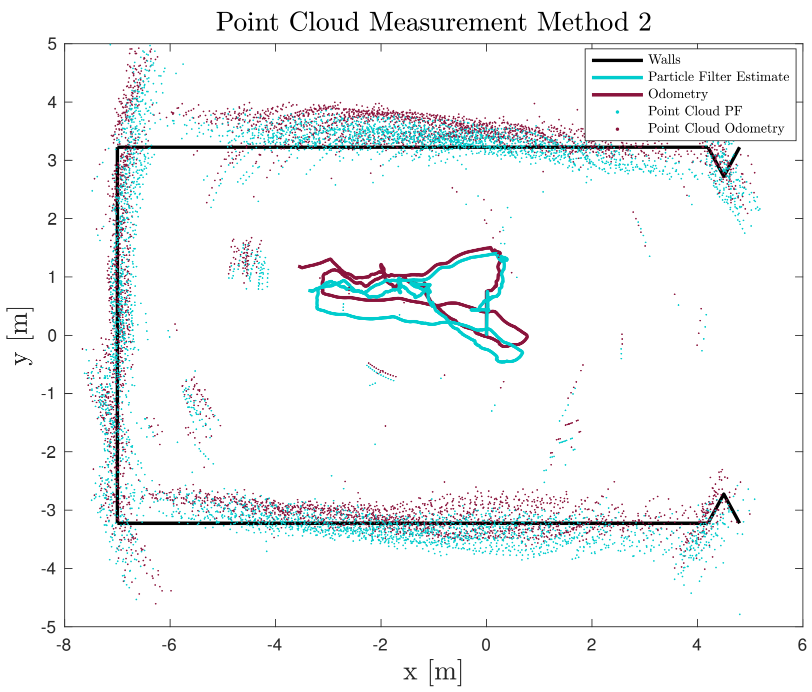


Figure 7.9: Sonar point cloud from position tracking with the measurement method 2.

7.2.2 Global Localization Problem

The measurement method 1 fails to localize the robot as the uncertainty of the initial heading increase. It is therefore disregarded when looking at the global localization problem. Only results from measurement method 2 will therefore be represented in this section.

Method 2

The particle filters ability to localize the underwater vehicle without any knowledge of the initial position of the vehicle is tested. This is done by initializing the particles from a uniform distribution covering the whole area of the water tank in the lab. The initial particles are therefore spread out over $260[m^2]$, and the particles initial heading is drawn uniformly between $\psi \in [-\pi, \pi]$. The process noise is set to $\sigma_\nu^2 = \begin{bmatrix}0.04 & 0.02 & 0.008\end{bmatrix}^T$ and the measurement noise $\sigma_{hit}^2 = 0.35$. In addition to this, the particle filter algorithm was set to only resample 80% of the particles, and add 20% new particles drawn from a uniform distribution $\pm 5[m]$ in x, $\pm 3[m]$ in y and $\pm\pi$ in heading around the position of the particle with highest weight, when the maximum weight of the particles was below $w_{max} < 2.5\frac{1}{N}$.

The result when using 1000 particles are shown in figure 7.10, where the particle filter estimation is plotted in light blue and the positions of the estimates where additional particles are added are shown with green marks. All the particles are plotted as dark blue dots for every 50th timestep, and one can clearly see that new particles are added at the left side of the tank due to the higher density of random particles in this area.

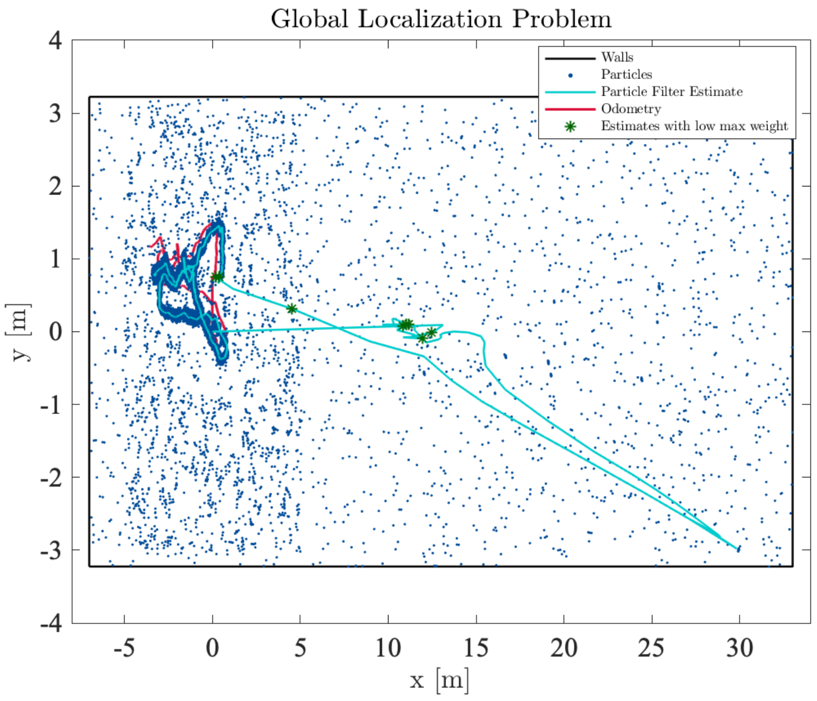


Figure 7.10: Global localization problem in Lab. The light blue line shows the particle filter estimate, and the dark blue dots are all the particles plotted for every 50th timestep. The green marks shows at what particle filter estimates the particle filter adds new samples around the particle with the highest weight.

Figure 7.11 shows the resulting sonar point cloud for the global localization problem for both the odometry estimate (red) and the particle filter estimate (light blue).

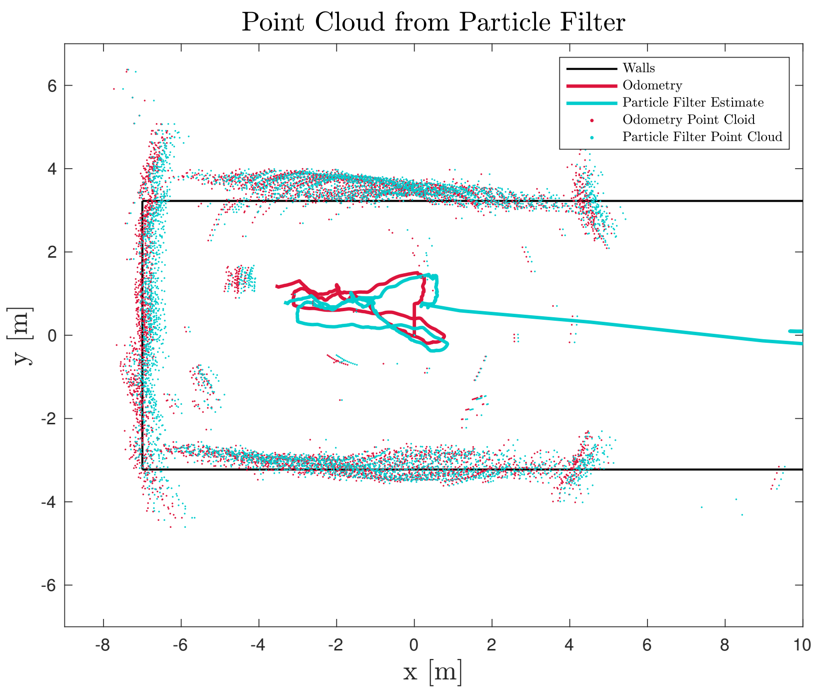


Figure 7.11: Point cloud from the sonar measurements for the odometry estimation and the particle filter estimation for the global localization problem.

The global localization problem does not convergence for every run. In order to evaluate the performance of the algorithm, the convergence rate is given in table 7.2. The simulation was run 50 times for each number of particles, with the particles spread out over the whole area of the tank.

Particles (N)	**Convergence**	**Percentage**
800	30	60 %
1000	36	72 %
1500	39	78 %
2000	41	82 %

Table 7.2: Convergence rate for a different number of particles for measurement method 2 with particles spread out over the whole area of the tank. 50 simulations were conducted for each number of particles.

In order to evaluate the performance of adding extra particles when the maximum particle weight is lower than a selected threshold, the simulation is also done for a standard particle filter which does not include this step. The percentage of convergence for 50 simulations with the standard particle filter is shown in table 7.3.

Particles (N)	Convergence	Percentage
800	3	6 %
1000	9	18 %
1500	12	24 %
2000	14	28 %

Table 7.3: Convergence rate for a different number of particles for the standard particle filter without adding random particles. The initial particles are spread out over the whole area of the tank. 50 simulations were conducted for each number of particles.

Due to the symmetry of the water tank, the particle filter converges to the wrong side of the water basin when it fails. This is shown in figure 7.12. The corresponding particle cloud is given in figure 7.13, where it is shown that the particle cloud from the particle estimate (light blue) still is consistent with the walls of the basin. The only object in the map input to the particle filter algorithm that differentiates the two sides of the basin is the triangular Qualisys cameras mounted on the walls which can be seen figure 5.6 of the map.

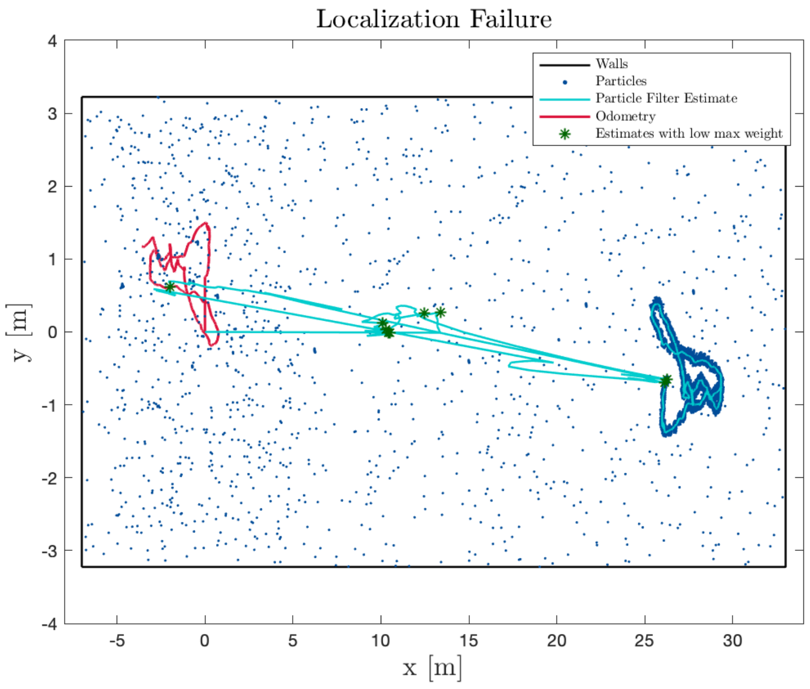


Figure 7.12: Results when particle filter estimate (light blue) converges to the wrong side of the water tank.

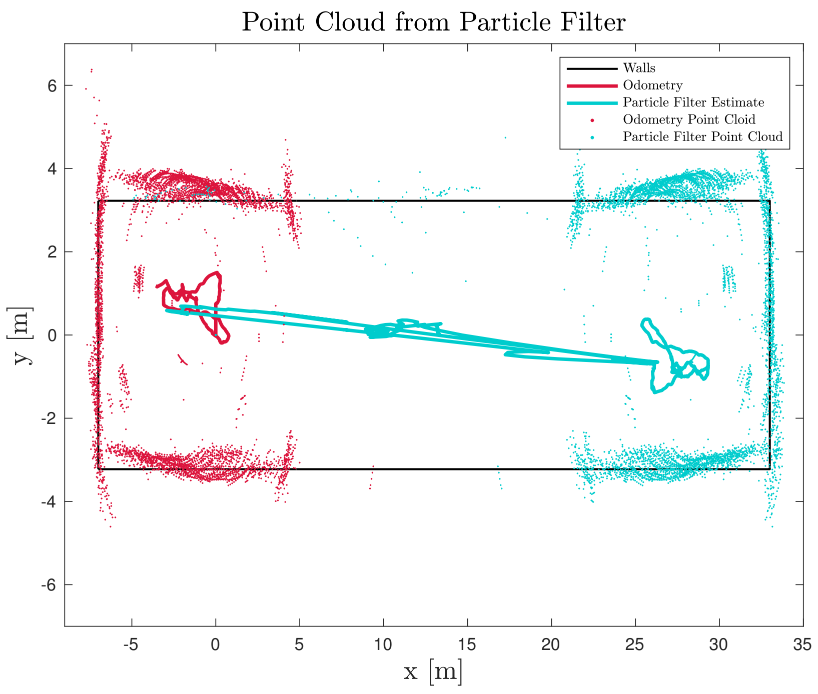


Figure 7.13: Point cloud from the sonar measurements for the odometry estimation and the particle filter estimation when localization failure occur.

Chapter 8

Discussion

In this section, the results presented in chapter 7 will be discussed. This will be divided into three parts, where the results from each dataset will be discussed separately, and then overall observations will be made.

8.1 Dataset 1

The particle filter was tested using the extensive dataset from the University of Girona, and the results were presented in section 7.1. Only measurement method 2 was tested using this dataset. First, the particle filters ability to perform position tracking was verified. The results from figure 7.1 showed the performance of the particle filter estimation compared to the dead-reckoning estimate. To see the improvement of the estimate, the translational error in x- and y-direction throughout the whole run was presented in figure 7.2. From this it can be seen that that the particle filter performs extremely well, and that the absolute translational error is close to zero compared to the dead-reckoning which is drifting and has an error of around $(35[m], 50[m])$ for x- and y-respectively at the end of the operation. Since the map used in the localization algorithm is constructed by the author and is not the true map of the area, this will affect the results and this is a source that will cause the error. On the other hand, it shows that the algorithm performs well with just an approximate map of the environment.

To investigate how the performance of the position tracking abilities of the filter varies with the number of particles used, 10 runs for each number of particles was tested and the average root mean square error for the runs was presented in figure 7.3. This showed that the difference between the accuracy did not vary a lot when the number of particles was more than 400. Since only 10 runs were performed, a 95% confidence interval was calculated for each of the configurations.

The particle filter was also able to successfully locate the vehicle without any knowledge of its initial position. The initial particles was spread out over $6825[m^2]$, and the filters estimation was shown in figure 7.4 with the translational error in figure 7.5. The error plot shows that the filter converges at around 500 [s], and performs well after convergence. The particle filter fails at some of the runs, and the convergence rate over 20 runs, for a different amount of particles, is shown in table 7.1. When using 3000 particles, the particle filter algorithm was able to perform satisfactory for all the 20 runs.

8.2 Dataset 2

The particle filters ability to perform position tracking was verified using both measurement methods. The figures showing the sonar particle clouds for both measurement methods, figure 7.7 and 7.9, proved that the resulting estimate was a good approximation of the vehicles true state. The mean distance of all points in the particle cloud was $0.22[m]$ and $0.24[m]$, for method 1 and method 2 respectively, compared to the mean distance of sonar point cloud for the odometry estimate which was $0.25[m]$. This is only a way to quantify the resulting estimate, and it would be more desirable to have a ground-truth to compare the results with. On the other side, when looking at the point clouds from the figures, both methods seem to provide a more accurate estimate than the odometry, which supports the numbers.

For the global localization problem, measurement model 1 failed to estimate the vehicle's position as the uncertainty of the initial heading increased. It was therefore not included in the section, and further development of the method must be conducted to obtain reliable results. The results from measurement model 2 was presented in section 7.2.2, where it was shown that the algorithm was able to solve the global localization problem and had an improved convergence rate when a number of random particles were added around the position of the particle with the highest weight as the overall maximum particle weight was lower than $w_{max} < 2.5\frac{1}{N}$. This threshold was found through trial and error and proved to be a good fit for all the different number of particles that were tested. In figure 7.10 the results from the global localization algorithm was presented, and the plot also included the position estimates at where the random particles were added. It showed that this occurred at timesteps where the estimated position was not sufficient, and it was able to recover and converge to the correct pose. To quantify the improvement of the particle filters performance when adding random particles, a convergence test was conducted for the proposed particle filter compared to the standard particle filter. This consisted of running the simulation 50 times for each number of particles, and calculate the convergence rates which are presented in table 7.2 and 7.3. For 2000 particles, the proposed particle filter converged 82% of the times, compared to the standard particle filter which only converged 28% of the runs. The reader should look at the tables for more results.

Due to the symmetry of the water tank, the wrong convergences converged to the right side of the basin which was shown in figure 7.13 with the resulting sonar point cloud in figure 7.13. The only object in the map that differentiated the two sides of the tank was the triangular Qualisys cameras that were mounted on the wall. Since the particle filter was able to converge to the correct solution 82% of the times, this shows the importance of having an accurate map of the environment and the accuracy of the range readings from the sonar.

8.3 Overall Observations

The particle filter algorithm shows promising results for both datasets. When compared to the standard particle filter it has a higher convergence rate, due to the new particles being added when the maximum particle weight is lower than a selected threshold. The measurement method 2 did not perform well when the uncertainty of the initial heading of the vehicle increased, and should therefore be improved. Measurement method 2 performed satisfactory for both datasets, and it does not require a lot of computational time as the values of the shortest distance to the wall is stored in a lookup table for each cell within the map. The fact that the map for the first dataset is only an approximate map of the environment, shows that the particle filter is still able to perform satisfactorily with only basic knowledge of the environment.

Chapter 9

Conclusion and Further Work

The first part of this chapter will draft a conclusion based on the results presented in this project, and the second part will give a short description of further work in regard to the presented topic of the thesis.

9.1 Conclusion

This thesis has provided a description and implementation of Monte Carlo Localization for autonomous underwater vehicles, consisting of a sonar-based particle filter. The author has successfully implemented a pose estimator, which is able to estimate the vehicles pose given sensor measurements available on the vehicle and a known *a priori* grid-map. The pose localization algorithm can solve both the position tracking problem and the global position problem and is able to recover in case of wrong convergence. The fact that the *a priori* map used in the first dataset is only an approximate map of the environment made by the author, which still gave satisfactory position estimate, is an important point as it is hard to obtain accurate maps of underwater environments and since movements of the ground over time might make changes in the existing maps.

To conclude, sonar-based particle filtering has shown promising results for both position tracking and localization of an underwater vehicle. Being further developed, it can be a key component for obtaining a robust and redundant localization system for autonomous underwater vehicles.

9.2 Further Work

The localization algorithm presented in this thesis is a passive localization algorithm. Future work would be to implement the algorithm for real-time localization of the vehicle during operation, and extending it to an active localization algorithm. An active localization algorithm requires access to the control system of the vehicle, in order to decide the best motion command which will minimize the time of convergence. In addition to this, this thesis has only considered the sonar sensor as a range finder, which does not utilize all the information from the sonar measurements, such as the echo intensities. Further work would be to develop a more accurate measurement model which includes the echo intensities that are measured by the sonar as well as the incident angle between the measurement and the object.

References

Bangare, S., Dubal, A., S. Bangare, P., Patil, S., 01 2015. Reviewing otsu's method for image thresholding. International Journal of Applied Engineering Research 10, 21777–21783.

Burgard, W., Fox, D., Hennig, D., Schmidt, T., 1996. Estimating the absolute position of a mobile robot using position probability grids. In: AAAI/IAAI, Vol. 2.

Caiti, A., Garulli, A., Livide, F., Prattichizzo, D., Jan 2005. Localization of autonomous underwater vehicles by floating acoustic buoys: a set-membership approach. IEEE Journal of Oceanic Engineering 30 (1), 140–152.

Collin, L., Azou, S., Yao., K., Burel, G., 2000. On spatial uncertainty in a surface long baseline positioning system.

Dellaert, F., Fox, D., Burgard, W., Thrun, S., 02 1999. Monte carlo localization for mobile robots. Vol. 2. pp. 1322 – 1328 vol.2.

Gustafsson, F., 2010. Statistical Sensor Fusion. Studentlitteratur.
URL `https://books.google.no/books?id=yd_2SAAACAAJ`

Karlsson, R., Gusfafsson, F., Karlsson, T., April 2003. Particle filtering and cramer-rao lower bound for underwater navigation. In: 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). Vol. 6. pp. VI–65.

Ramesh, N., Yoo, J., Sethi, I., 11 1995. Thresholding based on histogram approximation. Vision, Image and Signal Processing, IEE Proceedings - 142, 271 – 279.

Ribas, D., Ridao, P., Neira, J., Tardos, J. D., Oct 2006. Slam using an imaging sonar for partially structured underwater environments. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5040–5045.

Sandøy, S., 2019. personal communication.

Silver, D., Bradley, D., Thayer, S., Dec 2004. Scan matching for flooded subterranean voids. In: IEEE Conference on Robotics, Automation and Mechatronics, 2004. Vol. 1. pp. 422–427 vol.1.

Smith, R., Self, M., Cheeseman, P., 1990. Estimating Uncertain Spatial Relationships in Robotics. Springer New York, New York, NY, pp. 167–193.
URL `https://doi.org/10.1007/978-1-4613-8997-2_14`

Storkersen, N., Kristensen, J., Indreeide, A., Seim, J., Glancy, T., 02 1998. Hugin - uuv for seabed surveying. Sea Technology 39, 99–104.

Thrun, S., F. D. B. W. D. F., 2001. Robust monte carlo localization for mobile robots 128, 99–141.
URL `https://doi.org/10.1016/S0004-3702(01)00069-8.`

Thrun, S., Burgard, W., Fox, D., Arkin, R., 2005. Probabilistic Robotics. Intelligent Robotics and Autonomous Agents series. MIT Press.
URL `https://books.google.no/books?id=k_yOQgAACAAJ`

Øen Gustavsen, H., 2018. Proposal and comparison of an exogenous kalman filter and a particle filter for use with rov thruster models. Norwegian University of Science and Technology Department of Marine Technology, Norway.