FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Mapeamento e localização subaquática em mapas densos

Underwater mapping and localization in dense maps

Paulo Miguel Alves Gonçalves

MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Orientador: Prof.Dr. Bruno M. Ferreira

Coorientador: Prof.Dr. José C. Alves

29 de julho de 2022

Resumo

A área da engenharia contribuiu para o desenvolvimento de técnicas e ferramentas para serem aplicadas em ambientes submarinos. A robótica é implementada para fins militares, científicos e industriais para executar tarefas debaixo de água. Umas das ferramentas mais utilizadas é o veículo submarino autónomo (AUV). Este veículo tem como função operar um conjunto de tarefas de forma autónoma. Tarefas submarinas requerem normalmente o uso de sensores para obter informação do ambiente que rodeia o AUV. Em ambientes submarinos, sinais acústicos são os preferidos devido à longa distancia de propagação em ambientes subaquáticos em relação a outros tipos de sinais. No entanto, a presença de ruido no ambiente dificulta o uso de sinais acústicos. Nesta dissertação, é usado um AUV com um sonar instalado para obter informação do ambiente. O objetivo é desenvolver um sistema que seja capaz de filtrar informação útil do sonar para mapear o ambiente que rodeia o AUV e localizar este no mapa estimado. A informação obtida por o sonar é filtrada usando o algoritmo SOCA-CFAR, onde o output é alimentado para um algoritmo que extrai a primeira característica com o menor alcance dentro de um scan do sonar. Para o mapeamento do ambiente, é usado o método Octomap, onde os voxels são definidos com uma probabilidade de ocupação que são atualizados em cada ciclo. Estes voxels são considerados como volumes livres, ocupados ou desconhecidos no mapeamento. Para estimar a posição do AUV dentro do mapa, é usado um filtro de partículas (PF). Para o desenvolvimento deste sistema, foi feita uma séria de medições num tanque de água retangular, onde o AUV estava em repouso e em movimento.

Abstract

The engineering area contributed to the development of techniques and tools to be applied in underwater environments. Robotics is implemented for military, scientific and industrial purposes to execute underwater tasks. One of the tools more utilized is the autonomous underwater vehicle (AUV). This vehicle aims to operate a set of tasks autonomously. Underwater tasks usually require the use of sensors to obtain information about the AUV's surroundings. In underwater environments, acoustic signals are the favourite due to propagating long distances compared to other signals. However, the presence of noise in underwater environments complicates the use of acoustic signals. In this dissertation, it is used an AUV equipped with sonar to gather information. The objective is to develop a system capable of filtering useful information by the sonar to map the surrounding environment and localising the AUV in the estimated map. The data obtained by sonar is filtered using a SOCA-CFAR algorithm and it feeds to an algorithm that extracts the range to the first feature within a sonar scan. To map the environment, it is used the method Octomaps, where voxels are defined with a probability of occupancy that is updated each cycle. These voxels are defined as empty, occupied, or unknown volumes. A Particle Filter (PF) is used to estimate the AUV's position within the map. For the development of the system, a set of measurements was done in a rectangular tank where the AUV was standing still and in motion.

Acknowledgements

This dissertation is the beginning phase of the final chapter on what it was my five years journey as a student of electrical and computer engineering. There have been five years of experiences and hard work that will always be remembered throughout my life. This chapter will end at the end of this school year where it will begin a new part of my life which will start my professional career.

First of all, I want to thank both my supervisors and INESC researchers, Bruno Ferreira and José Alves, for giving me the opportunity of working in this area and for the continuous mentorship given in this dissertation.

To my family, especially my mother that raised me as a divorced mother, portraying as a mother and father at the same time, that gave everything she could so I could be here fighting for the best life I can have. I have to thank my brother as well for supporting me and for giving me hope and strength to continue to advance throughout these years in the toughest times.

To all my friends, I thank the old ones that followed my journey for the support and the new ones I made these five years for all the memories created.

Lastly, I want to thank DGES and the town hall of Barcelos for providing me with the scholarships that supported all the financial needs that I needed to study at FEUP as a student of electrical and computer engineering.

Paulo Miguel Alves Gonçalves

"You have competition every day because you set such high standards for yourself that you have to go out every day and live up to that."

Michael Jordan

Contents

List of Figures

List of Tables

Abbreviations and Symbols

2D	Two Dimensions
3D	Three Dimensions
AUV	Autonomous underwater vehicle
CFAR	Constant false alarm rate thresholding
DR	Dead Reckoning
DVL	Doppler Velocity Log
EKF	Extended Kalman Filter
FPGA	Field-programmable gate array
GPS	Global Positioning System
GPU	Graphic Processing Unit
INS	Inertial Navigation System
IMU	Inertial Measurement Unit
IPC	Interactive Closest Point
KF	Kalman Filter
LiDAR	Light Detection And Ranging
LBL	Long Baseline
MSIS	Mechanical Scanning Imaging Sonar
MBES	Multi-beam echo sounder
PF	Particle Filter
SIFT	Scale Invariant Feature Transform
SBL	Short Baseline
SLAM	Simultaneous Localization and Mapping
SBES	Single-beam echo sounder
SFB	Single Fixed Beacon
SHAD	Small Hovering AUV with Differential actuation
SONAR	Sound Navigation and Ranging
SEIF	Sparse extended information filter
SURF	Speeded-up Robust Feature
ToF	Time of fight

Chapter 1

Introduction

1.1 Context

The planet Earth can also be called the "Blue Planet" due to the abundant water covering the Earth's surface. Autonomous underwater vehicles (AUV) are increasing in popularity and importance for the realization of tasks in underwater environments. Nowadays, tasks such as building bathymetric maps and geological surveys, underwater cave or deep ocean exploration and underwater rescue missions are heavily dependent on these vehicles for their success. The development of a complex navigation system is dependent on the AUV's capability to extract meaningful information from its sensors, to locate itself in the environment and to plan its next move to complete its objective.

With an AUV, it is possible to build a map of its surroundings to gather information about the surrounding environment. An estimation of the environment is an essential component for the planning process for other AUVs, itself, populated vehicles, and dive missions. At the same time, maps are essential for the localization of the AUV, since it needs a map to locate itself in the environment. However, for real-time applications, it is necessary to construct a map that represents accurately the environment while the AUV is travelling. Furthermore, building a map is only possible if the vehicle can locate itself. This situation arises a problem for localization and mapping, which is a circular problem where these two aspects depend on each other to be accomplished. For this specific situation, it is applied the Simultaneous Localization and Mapping method, also known as SLAM, which is a set of algorithms that can locate the AUV and build a map of the environment while it is travelling. This breaks the circular dependence loop where an estimation of a previous map is not needed for these algorithms. Nevertheless, the development of a SLAM method is the most complex and difficult part of underwater applications. The improvement of these algorithms in terms of overall performance, memory required, processing time and power efficiency have been an activity with high priority for electrical and computer engineers that are integrated into this area of research.

1.2 Motivation

According to the authors in [36], AUV localization and navigation "*is particularly challenging due to the rapid attenuation of the Global Positioning System (GPS) and radio-frequency signals.*". Above the surface, most of autonomous systems rely on spread-spectrum communications and global positioning. However, they are unreliable due to the signals propagating short distances underwater. The acoustic signals are the solution, in this case, however, they still have many obstacles that need to be surpassed. Problems like the low bandwidth and low data rate, high latency, high variation of the sound speed due to different temperatures and salinity levels of the water and the multipath transmissions that could cause noisy measurements. Therefore, the development of a map building and localization algorithms are needed.

The developed system has to filter the readings gathered by the sonars to discard noisy and wrong measurements. Along with the development of algorithms that create and update a map, the result of the filtered measurements is required to be able to construct an accurate and readable estimation of the environment. Combining this with the pose estimation, it is possible to know where the AUV is travelling without using an external vehicle or communicating with a surface system.

1.3 Objectives

The work proposal of this dissertation is the development of an autonomous underwater map builder using data collected by a sonar, more specifically, a mechanical scanning imaging sonar (MSIS). This sonar is installed in a compact torpedo-shaped AUV (SHAD) which is used as test vehicle in this study.

The objective is to study and understand the essential concepts, methods, techniques, and algorithms that can be used, compare their performances and what are the most appropriate for this case. Furthermore, it is expected that the system developed can extract useful information about the environment to construct a map and estimate the position of the AUV on that map.

Chapter 2

State of the Art

To understand the current state of the art in an autonomous map building, it is done a research focusing on the necessary aspects around the task at hand. First, it is done a study on the AUV's localization techniques that can be used to locate the vehicle. This section also explores the different types of sensors that can be used to localize the AUV. A more detailed study is done about the sonar sensors, exploring their characteristics and the different types of sonars that exist today. Furthermore, it is presented the different methods of environment representation, describing their approach for the mapping process. It is done an exploration of the different categories that a SLAM can be classified and a study of loop closure, data association and scan matching methods that are essential concepts in SLAM. Furthermore, the SLAM algorithm can be divided into different types depending on how the data is gathered and how it is filtered.

2.1 Underwater Localization Techniques

First, when the objective is developing an underwater map-building algorithm for an AUV, it is important to develop an accurate localization and navigation system. This is an essential step since the localization allows to match readings obtained by the sensors with the existing estimated map.

It is well known that the global positioning system (GPS) and radio communications are not reliable options in underwater situations. This is due to these type of signals only propagating short distances ([36]). Therefore, to perform the real-time estimation of the AUV's navigation, other more expensive sensors are usually employed. Acoustic based sensors and communications have a better performance in underwater environments. However, acoustic based sensors and communications suffer some inconveniences that have to be accounted for, such as ([20];[36]):

- Small bandwidth that forces the implementation of techniques to share information;
- Low data rate, which restrains the number of data that can be transmitted;
- They depend on sound speed, which varies due to the salinity and temperature levels of the water;

- Attenuation of the sound wave, which restricts the usable information at a maximum range;
- Multipath transmissions that are generated due to multiple reflections of the sound waves in the upper and lower boundary.

According to Leonard et al.[24] and Paull et al.[36], these sensors and techniques can be divided into one of these categories:

- Inertial Navigation: Uses accelerometers, gyroscopes, and Doppler Velocity Log (DVL) and/or dead reckoning techniques (DR) to increase the accuracy to propagate the current state.
- Acoustic transponders and modems: Technique that is based on measuring the time of flight (ToF) of acoustic signals coming from beacons or modems. Runs trilateration and triangulation algorithms to find the vehicle's location.
- Geophysical: Localization is estimated based on the information gathering of external landmarks by sensors, that can detect and identify features in the surroundings.

2.1.1 Inertial Navigation

Inertial Navigation System (INS) is a system that receives measurements to estimate and track the position and orientation of the vehicle, compared to a known starting point ([4]; [5]).

Dead Reckoning (DR) is a localization technique, that consists of localizing the AUV without any valid measurements from the acoustic sensors or transponders. DR techniques are a way to navigate the AUV to the next position only based on its orientation and motion. However, there is one problem with DR techniques due to cumulative errors ([21];[36]). The cumulative phenomenon causes the unbounded growth of the error in the position estimation with distance travelled. This is the reason why an INS helps the pose estimation calculated by purely model-based DR techniques, due to providing measurements of the motion of the vehicle and orientation each cycle to correct the estimation. Still, even with an INS, the error is growing (but slower) while travelling and still is unbounded since the sensors are also not perfect on their measurements.

For INS, some sensors can be used to provide measurements for the position estimation presented in the table 2.1. INSs are usually complemented with sensors for localization and motion corrections such as the Global Positioning System (GPS) which estimates the position via the TOF of signals from satellites and the Doppler Velocity Log (DVL) which determines the velocity of the vehicle.

2.1.2 Acoustic Transponders and Modems

The estimation of the AUV's location can be accomplished by using the support of external artificial landmarks. They send or reflect acoustic signals where the estimation is based on the TOF of those signals.

Table 2.1: Common sensors used for INS

Sensor	Description
Accelerometer	Provides the linear acceleration based measurements of the force acting on the object
Gyroscope	Device used for measure or maintain the orientation and angular velocity based on angular rates
Inertial Measurement Unit (IMU)	Sensor that is a combination of accelerometers and gyroscopes used to estimate the object's orientation, velocity and gravitational forces

In underwater situations, localization using acoustic transponders and modems can be achieved by common methods such as Short baseline (SBL) and Ultra short baseline (USBL) that uses beacons placed in the hull of the AUV, and Long baseline (LBL) that places beacons around the seafloor ([21];[36]).

2.1.3 Geophysical

Geophysical techniques are dependent on sensors being capable of identifying and detecting features in the environment to estimate the localization.

According to the authors in ([36]), the geophysical methodologies can be categorized as:

- Magnetic: Utilizes magnetic field maps for localization. Although this method is not very common due to being easily interfered by other electromagnetic fields, the past years have been applications and studies for AUV navigation like in [23] or in [37].
- Optical: Uses mono or stereo cameras to capture images of the seafloor and match them to a pre-built map or estimated one to locate the AUV. The disadvantage is, that when the water where the AUV is travelling is not clear or at high depths, light does not reach the seafloor which makes it difficult the estimation of the localization by this method.
- Sound Navigation and Ranging (Sonar): This method is most common in underwater navigation and was developed specifically for that type of environment. The reason is that sonars are sensors that work with acoustic signals. As of today, acoustic signals are the ideal option to be employed in underwater missions.

2.2 Sound Navigation and Ranging

The Sound Navigation and Ranging (sonar), is developed to operate at certain frequencies that depend on the range and the resolution desired for underwater applications. Furthermore, sonars can be considered active or passive ([36]). Passive sonars are just listening devices that read the sound propagated in the water. Active sonars generate sound waves and read their echoes when they are deflected by the surface or objects underwater. Sonars can be characterized as imaging sonars and ranging sonars ([36]). Imaging sonars insonify a swath of the seabed to gather

measurements. Ranging sonars are more focused on measuring distances between the transducer and the seafloor or objects/obstacles. Sonars can also be divided by their type, as is shown in the table 2.2 ([20];[36]).

Table 2.2: Common sonar types

Sonar type	Description
Single-beam echo sounders (SBES)	SBES just emit a single beam to a single point beneath the transducer. Commonly used to measure the depth of the ocean beneath the vehicle. This type of sonars has the problem of inaccurate measurements due to the spread-out behaviour of the beam, which could lead to receiving an echo that does not come from directly below
Multi-beam echo sounders (MBES)	MBES uses an array of transducers to illuminate a narrow swath that is perpendicular to the path of the AUV and calculates the distances with the TOF of the signals. This type of sonars are very accurate but they are only appropriate for boats/ships or heavier AUVs
Sidescan	A type of sonar that emits a sound wave to the seafloor, where there are projectors on the opposite sides of the AUV (front and back) and the swath illuminated is perpendicular to the AUV's trajectory.
Forward scan	Forward scan is the same concept but the sonars illuminate parallel to the seafloor, with the objective of detecting an object in front (or back) of the AUV
Mechanical Scanning Imaging Sonar (MSIS)	A single beam sonar, that scans a certain area in one cycle. This beam can just rotate for a few degrees or can rotate the full 360 degrees. This type of sonars are very compact and cheaper compared to multi-beams or sidescan sonars, which are an appropriate to use in lighter AUVs
Profiler	Special type of sonar that works at low frequencies. Profiler can penetrate the seabed, detecting buried features with good resolution. This type of sonars are normally used to search for untapped materials underwater or to maintain underwater piping

Sonars are characterized by their shape, where they have a certain width and an angle that depends on the sonar itself.

The Fig. 2.1 is a graphical representation of a sonar beam. Unlike a laser beam, a sonar beam is characterized by its shape. Its shape is defined by an horizontal angle, $\delta_{horizontal}$, and a vertical angle, $\delta_{vertical}$ ([25];[43]). The data gathered by a sonar beam is an array of intensities provided by the echoes from the environment. This array is divided into bins that relate intensity values with their corresponding distance. Knowing the bin length of the sonar, it is possible to determine the distance of each intensity value by multiplying its index in the array with the bin length of the sonar. On the other hand, a single sonar beam is not capable of knowing exactly the shape of the object being detected, since there is no information on the angle within the sonar beam. Therefore, a sonar model must be applied to recreate the closest representation of reality.

The sonar model applied in this dissertation considers all points within the sonar beam with the same distance as a certain intensity. As is shown in Fig. 2.2, an obstacle is detected in the sonar beam and all the points within the same range as the obstacle are considered occupied points. This

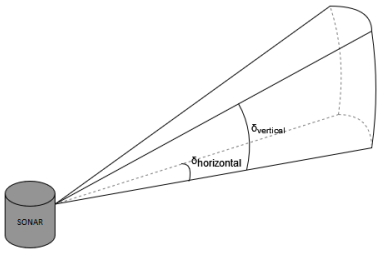


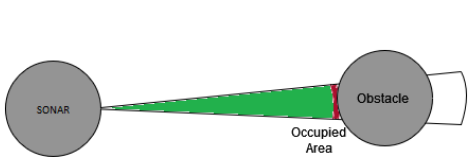
Figure 2.1: Graphical representation of a sonar beam characterized by its horizontal and vertical angles

model allows to accurately define the empty space surrounding the sonar only knowing the range of the detected obstacle.

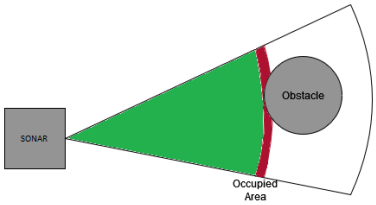
This study uses an MSIS that can do full revolution scans. In this case, a full 360º cycle generates an acoustic image that represents the intensities of the echoes from the surroundings of the AUV. The Fig. 2.3 is an example of an acoustic image generated by an MSIS sonar. This image is composed of k sonar beams where each beam contains the array of intensities divided by bins. This acoustic image can be filtered by a threshold algorithm and applied in a feature extraction algorithm to obtain the obstacles detected by the sonar and filter out noisy measurements. This is one way of obtaining information with raw data using a sonar to build a map of the environment.

2.3 Environment Representation

Underwater maps are useful to know how the environment is shaped and they can be used as a point of reference for automatic vehicles to execute various tasks underwater. However, if the area is unknown, an AUV that is capable of constructing a map is needed. In terms of information, the map created can be a dense or sparse representation. A dense map is a detailed map of the environment with a large number of features represented. According to [42], the more landmarks a map contains, the more accurate the algorithms that are responsible for localization are since

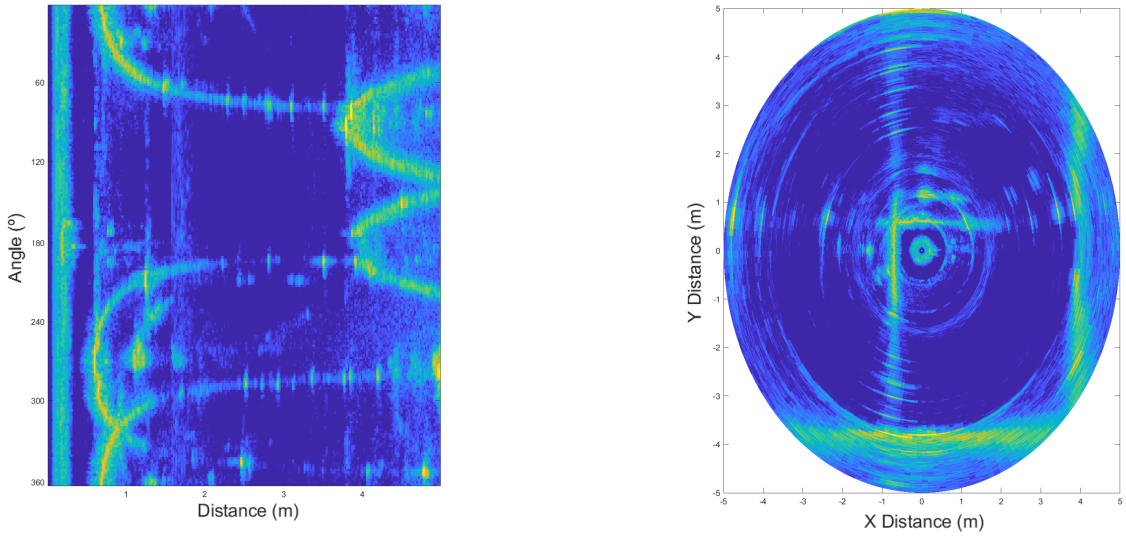


(a) Top view of sonar beam



(b) Front view of sonar beam

Figure 2.2: Sonar model



(a) Acoustic image in polar coordinates

(b) Acoustic image in cartesian coordinates

Figure 2.3: Example of an acoustic image obtained by a MSIS

their solutions are more precise. However, it requires a high memory to process a dense map. A sparse map is a sample of the environment with the key features represented in the map. This type of map requires lower memory to process than a dense map, however, contains less information about the environment which decreases the accuracy of the representation.

2.3.1 Point Clouds

A point cloud is a set of discrete points represented in a defined dimensional coordinate that contain unique features which form a sample of the environment. In a 3D space, a point in a point cloud normally is defined by its coordinates (X,Y,Z) to a known origin point or these may have additional information such as RBG values and intensity values ([2]). According to [28], point clouds are popular with applications using laser range scanners such as LiDAR scanners and RGBD cameras. In general, these scanners provide an unorganized point cloud that represents the surface of the surroundings.

2.3.2 Occupancy Grid and Octomaps

The occupancy grid consists of the division of a 2D/3D space into a 2D/3D array that contains cells which are classified as empty or occupied spaces. A binary occupancy grid is the simpler method that just defines the cells into two values: 1 if it is empty or 0 if it is occupied. The other popular method is the probabilistic occupancy grid introduced by [30], where the cells are defined by a probability of occupancy. Since motion and environment measurements obtained by the sensors are not perfect, all cells in the probabilistic version can be classified with an uncertainty value. Both occupancy grids are presented in [7].

Figure 2.4: Point cloud example. Adapted from [26]

In general, 3D occupancy grids are composed of voxels. A voxel is a 3D cubic representation of a detected volume of an object. Voxels can be looked at the same way as pixels for 3D space, where the detected obstacle is represented by a cell. The normal procedure is to recursively divide the 3D space into 8 equal cubic volumes until the voxel reaches a predefined minimum side length. These voxels with the predefined minimum side length define the resolution of the map ([19]). Voxels are applied in various types of tasks due to their simplicity and are effective to represent a volumetric object which is appropriated for building a map but requires high memory to work with great resolutions ([29]).

Despite this technique being commonly employed in autonomous vehicle applications, it has the limitation of the defined map size being fixed. This is a problem when mapping an unknown area due to the dimensions of the environment being unknown, which makes this technique unreliable. The heavy computational load is also one problem that makes the 3D application less appealing which forces additional development to accelerate the process ([13]).

Nevertheless, a variation of occupancy grids was developed to address the limitation of the fixed-sized map which is the Octomaps. According to [19], Octomaps are an efficient probabilistic 3D mapping based on Octrees. This technique uses a 3D occupancy grid mapping approach that

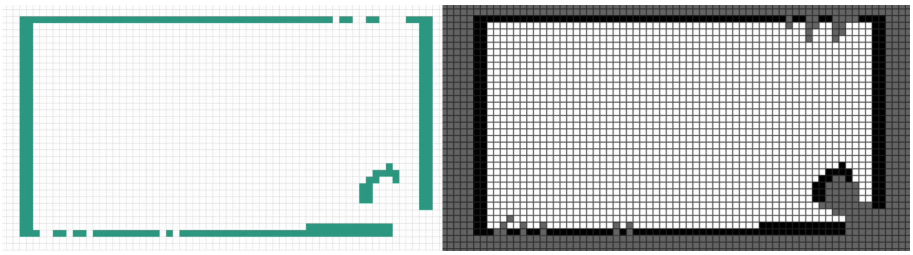


Figure 2.5: 2D vision of occupancy grids: binary (left) and probabilistic (right). Adapted from [40]

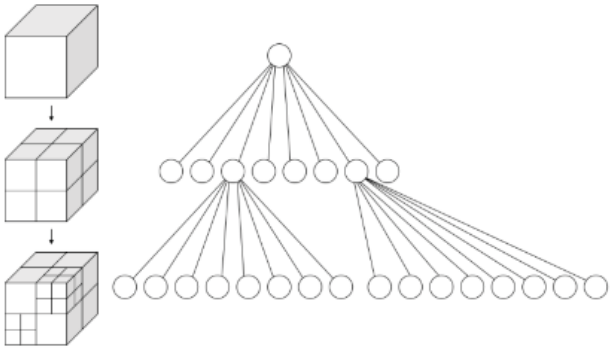


Figure 2.6: Octomap (left) and octree (right). Adapted from [9]

can model the environment without any prior knowledge.

Fairfield et al. [11] and Hornung et al. [19] define an octree as an expandable hierarchical data structure of the spatial division of the 3D space. It resembles a tree-based diagram where the nodes represent a sub-space represented in the map and the last ones, which are usually voxels, are classified based on the certainty of their occupancy. The inner nodes (larger cubic sub-spaces) are classified either as the average of the occupancy or it is equal to the maximum occupancy of their children. The octree approach eliminates one of the disadvantages of occupancy grids, which is the fixed dimensions of the map. By being an expandable tree, its dimensions are dynamic which can be applied in the real-time mapping of an unknown area. It can maintain its nodes accordingly to the sensor measurements and cut at any level if the information present on the tree is not necessary (like eliminating dynamic features detected). This ability to dynamically change the tree also compacts the information of the map in terms of memory and the usage of space on the disk.

2.3.3 Topological Based Maps

The author in [16] presents that topological-based maps model the environments as a graph-like representation, where the nodes represent distinctive places and model the relations between the nodes. This implementation provides a compact representation of the environment that only displays certain places of the environment.

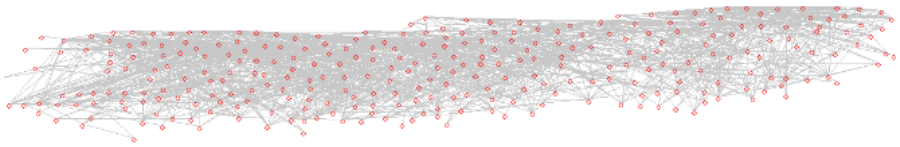


Figure 2.7: Topological map example. Adapted from [8]

This type of map is very effective in larger areas of exploration due to requiring low memory. It can be useful for path planning and there is no need for precise localization since the AUV could just follow the flow from the topological map ([3]). However, this technique struggles to map the volumetric shapes and is not reliable for tasks that require high accuracy due to being more of an abstract representation of the environment.

2.4 Simultaneous Localization and Mapping

The task of localizing and mapping an unknown area can be described as a chicken egg or problem: to know the vehicle's location it is needed a map of the world, but at the same time, to build a map it is needed a pose estimation to know where the scans take place in. Simultaneous Localization and Mapping is a solution to this problem. Also known as SLAM, it is an algorithm that receives measurements done by the sensors, estimates his pose and builds a map at the same time while the AUV is travelling.

SLAM can be categorized as online or full/offline ([18]). An online SLAM does an estimation of the pose and the map in each cycle while the full/offline SLAM estimates all the poses and the map of the entire path using the set of measurements obtained during that trajectory.

Additionally, SLAM algorithms can also be divided as feature-based and view-based ([10]; [18]; [36]), whereas in feature-based SLAM, the position of the features is estimated and maintained in the space state with the localization of the vehicle. In view-based SLAM, a set of vehicle poses is estimated using vehicle states composed of locations at the time of the observations.

2.4.1 Scan Matching

Scan matching process is an important step in building a map. The sensors used to gather information about the environment and vehicle's motion and orientation will always have uncertainty.

This uncertainty leads to errors in the measurements and errors in the estimation of the localization, resulting in mapping errors. These errors and uncertainties cause the misalignment of the map and distorted scans done by the sensors, which causes the generation of a map that does not make sense. Therefore, in every map builder, it is required an algorithm that matches the measurements and aligns the scans done in each cycle, so it constructs a useful map. The objective of scan matching algorithms is to align the scans done in each cycle to construct an estimation of the environment that makes sense ([31]; [33]).

There are different approaches of scan matching algorithms that tackle them and as it is presented in [5] and [27], these approaches can be categorized as:

- Feature-based techniques

  The feature-based technique is a scan matching algorithm that tries to match the different scans through the features they detected. This means that if two different scans detect the same feature, this algorithm will connect these scans to make them aligned. These features could be corners, edges, arcs or even shapes detected by the sensor. Some techniques such as 3D Harris corner detector, Scale Invariant Feature Transform (SIFT) and Speeded-up Robust Feature (SURF) offer feature and description detection that contributes to a scan matching algorithm.

- Compact data methods

  Compact data methods utilize mathematical properties from range measurements to match the scans. It can use histograms, motion fields or eigenvectors in the method. However, these measurements are very sensitive due to the presence of dynamic objects in the environment, and they are affected by the noise.

- Point matching techniques

  Point matching techniques are the most used in scan matching. The matching is done directly by establishing correspondences between points from two different scans obtained. This technique is appropriate for real-time systems with motion estimation and dynamic surroundings because they use the raw data obtained by the sensors. This technique is normally applied in maps with point cloud representation.

  The most applied point matching technique is Interactive Closest Point (ICP): estimates a rigid transformation to reconstruct a 3D surface from different measures from the scans. It matches the closest points from the two scans, trying to estimate the transformation (rotation and translation) done in one point of a scan to be in the same position as the matched point in the other scan, minimizing the error. Some other approaches are used such as gradient computation (based on cost functions), evolutionary programming and genetic algorithm (the last two based on stochastic components), but ICP has the advantage of being simple and effective.

2.4.2 Data Association and Loop Closure

According to Aulinas et al. [1], common data association methods are to match two successive scans or to close a loop of a long trajectory which makes the vehicle return to the starting point of the trajectory, obtaining the same feature of the map or previous localization in two separate scans. However, this process needs to have a significant level of robustness for the feature correspondence, since underwater scenarios are known for distortions and noise problems.

Another problem in the SLAM algorithm that must be addressed is the loop closure problem that is related to the data association. As was mentioned before, data association can be solved by closing a loop, which makes the vehicle take separate scans of the same feature or localization. However, due to accumulated errors in the estimation of the current and previous positions and orientations of the AUV or in the localization of the features during the trajectory, the loops that are formed are not properly closed.

2.4.3 SLAM approaches

The existence of various types of SLAM algorithms is vast. Since this algorithm is essential for every type of autonomous vehicle that maps an unknown area, SLAM algorithms have been studied and improved greatly over the years. Nevertheless, these improvements and variations are originated from common and popular SLAM algorithms ([18]; [36]).

- EKF SLAM: SLAM algorithm that implements an extended Kalman filter (EKF). Usually used in feature-based and online SLAM, EKF SLAM is one of the most common SLAM algorithms due to its simplicity and good results when features are very distinct in the environment. EFK estimates the location through the mean value and covariance where it uses state-space models (DR techniques and odometry) and observation models (sensor measurements) to result in a prediction and correct that prediction for the pose estimation of the vehicle and the landmarks detected.

- SEIF SLAM: The sparse extended information filter (SEIF) SLAM is an algorithm that implements an extended information filter. Usually employed in feature-based SLAM and it is an online SLAM. SEIF SLAM is similar to the EKF SLAM due to the filters implemented work in a similar manner. However, this SLAM filters and removes information that is considered "weak" (large covariances), maintaining only strong and Gaussian values in the information matrix.

- Fast SLAM: Fast SLAM is also one of the most popular SLAM algorithms used. This algorithm implements a particle filter, also known as the Monte-Carlo filter. Usually applied in feature-based SLAM and it is an online algorithm. Estimates the position of the vehicle and the map based on particles and it is possible to estimate non-Gaussian processes.

- Graph SLAM: This is an offline SLAM that stores raw measurements obtained by the vehicle and estimates all poses locations and the map, considering all the estimations as nodes

and linking these scans with constraints. Very effective and consistent in estimating but requires high computational power.

- AI SLAM: It is an online SLAM that is based on artificial intelligence and deep learning concepts. It is efficient due to imitating the way that animals' brains work but they required initial training.

- vSLAM: Visual SLAM (vSLAM) algorithms are based on utilizing a type of camera to detect features and build a map with them. Unlike the other examples, vSLAM is a broad algorithm that contains different types of algorithms such as MonoSLAM, RGV-D SLAM and PTAM. These types of SLAM algorithms are mostly employed for indoor applications (for example, automatic vacuum cleaners) and mapping of surface areas (autonomous driving).

2.5 Related Work

This dissertation serves to explore and contribute to this area of research and for that reason, there is a necessity to identify similar works that use the concepts above in practice. In this section, a general description of the works is done, highlighting aspects of interest.

Due to their simplicity and general good results, the EKF SLAM is employed in various works for underwater applications like [6], [34], and [41]. In [41] is developed a robust EKF SLAM, addressing some of its limitations and describing in-depth this implementation. [6] works with LBL and with EKF SLAM to estimate the localization and orientation of the vehicle. Furthermore, [34] constructs an underwater structure with a laser scan for inspection using EKF to estimate the vehicles and features positions. [11], [12] and [15] are examples of using FAST SLAM, an algorithm that works with any type of random distribution (unlike EFKs). [11] is a real-time SLAM using particle filters to explore underwater tunnels, using the FAST SLAM for the estimation of the vehicle poses. The article [12] uses FAST SLAM which defines walls as features to estimate the AUV's localization and construct a map. The work in [15] is a dissertation that utilizes a simple FAST SLAM where the mapping is done through a probabilistic attribution and update for each space in the occupancy grid. The pose Graph SLAM method is not as common as the last two but works such as [35] use a Graph SLAM for underwater exploration, where it uses the ICP method for scan matching to estimate the locations of the AUV in the trajectory and match all the scans done to form a map. In most of these projects are employed DVLs, a device very reliable but used for heavier AUVs, which is a problem for this dissertation that works with a light AUV.

For the environment representations, some of the previous works mentioned ([11]; [12]; [15]) implemented occupancy grids or the variation Octomaps which work with probabilistic methods of occupancy while using ray-tracing as sonar model, for the exception of [12] that uses a *Filter Chain* due to the walls being the only features considered. [35] represents the sonar measurements in form of point clouds paired with ICP for matching the scans.

Chapter 3

Mapping and Localization System

In this chapter, an overview of the developed system, the tools involved in the project and the environment where the measures were taken are presented. The underwater vehicle used is the SHAD AUV (Small Hovering AUV with Differential actuation) presented by Gonçalves et al. [17]. This vehicle is a micro-sized AUV shaped like a torpedo with 120 mm in diameter, 1.10 m long and it weighs 9 kg. Due to its size and weight, the AUV is equipped with an MSIS, more specifically, a mechanical scanning sonar (MSIS) that is mounted parallel to the floor to obtain information about the surroundings of the vehicle instead of the seafloor.

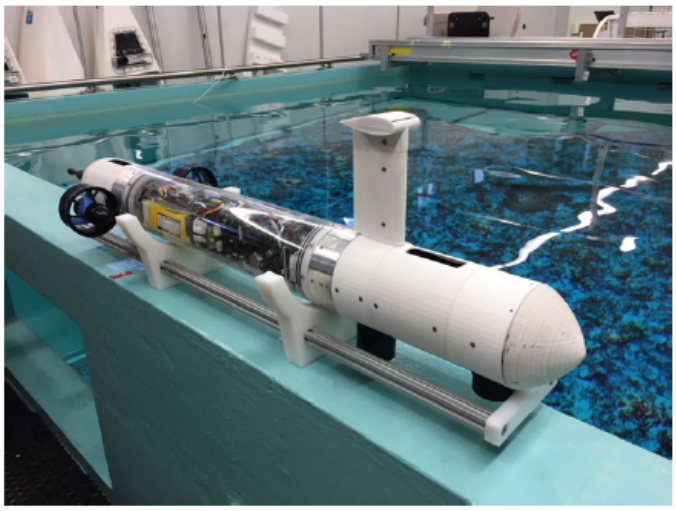


Figure 3.1: SHAD AUV used to obtain data that is used in this dissertation. Image adapted from [14]

This sonar is an ultra-compact imaging sonar that is centred on 700 kHz in operating frequency. Its beamwidth has 35° vertical and 3° horizontal, and its maximum and minimum range are 75 m and 0.3 m, respectively. The beam of this sonar can do a full 360° scan in one cycle that takes about 8.5 seconds to complete. Lastly, this MSIS has an approximately minimum range resolution of 7.5 mm and has four different mechanical resolutions: 0.45°, 0.9°, 1.8°, and 3.6°.

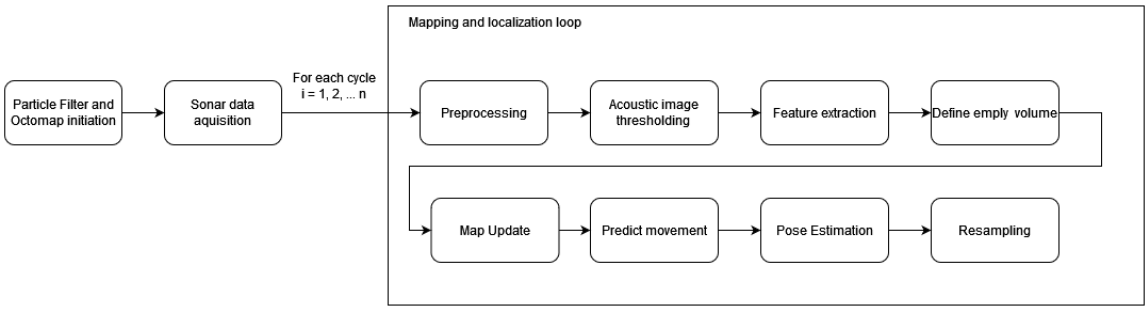


Figure 3.2: Block diagram of the system

The Fig. 3.2 presents a block diagram, portraying the overall architecture behind the solution developed. The system uses the data obtained from the sonar to build a map of its surroundings and locate itself in that environment. First, this system initializes a new map and creates the initial particles for the particle filter. The system collects the data given by MSIS each time it concludes a cycle, which is a full 360º scan. At this point, the system enters a loop for each cycle obtained. The measurements obtained are fed into a preprocessing task. This task filters the intensities that have a lower range than a constant *d0*. Typically, these intensities are vehicle reflections which are unwanted measurements for this study. Furthermore, the data is fed into a thresholding technique, where the intensity matrix is filtered to have only the desired intensities that have the potential of being a feature in the environment. The output is fed into a range of the first feature algorithm, an algorithm that finds the first valid feature in each step angle array to define the empty and occupied areas around the vehicle. With this result, a probability of occupancy is defined, where the empty area has a low probability, the occupied has a high probability and the remaining areas are considered unknown. With the information of what areas are occupied or not and their probability, they update the probability of the occupancy voxels of an Octomap, a grid occupancy map based on Octrees. Lastly, the system predicts the motion of the vehicle and applies this into the particles of the PF. The estimation of the localization is done and at the end, a resampling of the particles is done to estimate the localization for the next cycle.

In this project, the system is designed to read datasets gathered by an MSIS sonar. In general, datasets are an array that provides the instant time taken for the measurement, the horizontal angle of the measurement, the step distance which is the range resolution of the MSIS and an array of intensities.

3.1 Acoustic Image Thresholding Using Raw Data

Acoustic image thresholding is a crucial step in finding the desired features present in the data gathered by the sonar. The main goal of this algorithm is to filter the matrix of intensities, where if

they are under a certain threshold, they can be ignored. This threshold can be either fixed or defined by a function, for example, a mean or median function. Image thresholding is a frequent practice and various techniques can be applied. In this study, four types of techniques experimented with, and it is done a comparison of their results in a later chapter.

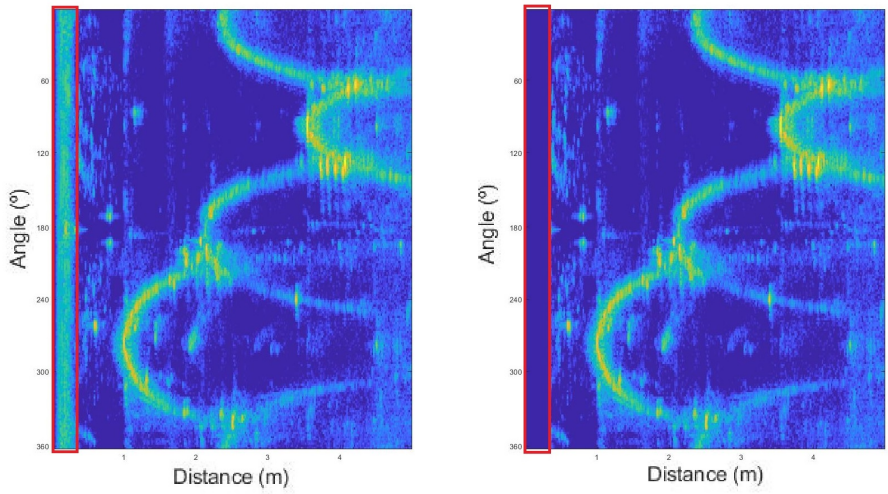


Figure 3.3: Example of an acoustic image with a red rectangle representing the intensities values ignored that have lower range than *d0*

Before applying the threshold techniques to the acoustic image, it is done a preprocessing task that filters the values that have a lower range than a constant *d0*. The data gathered used in this dissertation have high intensities near the AUV's position due to the vehicle reflections obtained by the sonar. In Fig. 3.3 shows an example of an input acoustic image on the left and the output after ignoring the intensities with a range lower than *d0* on the right.

3.1.1 Simple Threshold

The simple threshold technique defines a constant threshold for each acoustic image, where if the intensity value is higher than the threshold, maintains its value and if not, the intensity value becomes 0. This constant threshold is defined by a simple function. For example, it can be a percentage of the maximum intensity present on the matrix ([32]) or the mean of the matrix.

3.1.2 Otsu and Multi-Otsu Threshold

The Otsu threshold algorithm is a binary process that filters an image divided by its grey levels. Given an image with L greyscale levels with a total of pixels N, the spread of the pixels in each greyscale level is analysed. Furthermore, all greyscale levels, L, are potential thresholds, which is done a set of calculations to determine the most optimal threshold.

Considering the threshold being analysed as T, the image is divided into two classes, where the pixels within the greyscale levels {0,...,T-1} are in the background class and pixels within the

greyscale levels {T,...,L-1} are in the foreground class. The next step is to calculate the weight (ω_b, ω_f), mean, and variance of the two classes.

ω_b, ω_f can be obtained through the Eq. 3.1 and Eq. 3.2, respectively ([39]).

$$\omega_b = \sum_{i=0}^{T-1} \frac{n_i}{N} \tag{3.1}$$

$$\omega_f = \sum_{i=T}^{L-1} \frac{n_i}{N} \tag{3.2}$$

where n_i is the number of pixels present on the greyscale level i. To obtain the mean and variance of the two classes, it is done as follows:

$$\mu_b = \sum_{i=0}^{T-1} \frac{i \times n_i}{n_b} \tag{3.3}$$

$$\mu_f = \sum_{i=T}^{L-1} \frac{i \times n_i}{n_f} \tag{3.4}$$

$$\sigma_b^2 = \sum_{i=0}^{T-1} \frac{(i-\mu_b)^2 n_i}{n_b} \tag{3.5}$$

$$\sigma_f^2 = \sum_{i=T}^{L-1} \frac{(i-\mu_f)^2 n_i}{n_f} \tag{3.6}$$

where n_b and n_f are the number of pixels in the background and foreground, respectively.

These values are calculated with the purpose of determine the within-class variance, *WVar*.

$$WVar = \omega_b \sigma_b^2 + \omega_f \sigma_f^2 \tag{3.7}$$

After calculating all the within-class variance for all greyscale levels, the optimal threshold, in theory, will be the one that has the lower value for within-class variance, obtaining a binarized image where the foreground is considered as 1 and the background is considered as 0. For a faster approach to determine the optimal threshold, instead of calculating *WVar*, it is calculated the between-class variance,*BVar*, and the optimal threshold will be the one that has the higher *BVar* ([39]):

$$BVar = \sigma^2 - WVar = \omega_b \cdot (\mu_b - \mu)^2 + \omega_f \cdot (\mu_f - \mu)^2 \tag{3.8}$$

where $\mu = \omega_b \mu_b + \omega_f \mu_f$.

The Otsu thresholding only divides the image into two classes. However, it is possible to divide the images into more classes by applying the multi-Otsu thresholding. This technique follows the same concept as calculations of the Otsu method, but the image is divided into various levels with multiple thresholds instead of binarizing the image. The between-class variance for multi-Otsu algorithms is calculated as follows:

$$BVar = \sum_{i=0}^{N_{classes}-1} \omega_i \cdot (\mu_i - \mu T)^2 \tag{3.9}$$

where $N_{classes}$ is the number of classes defined and $\mu T = \sum_{i=0}^{N_{classes}-1} \omega_i \mu_i$.

This way, it is possible to filter an image by considering the values are above the desired levels, making this thresholding method more flexible than the Otsu method.

3.1.3 Constant False Alarm Rate Threshold

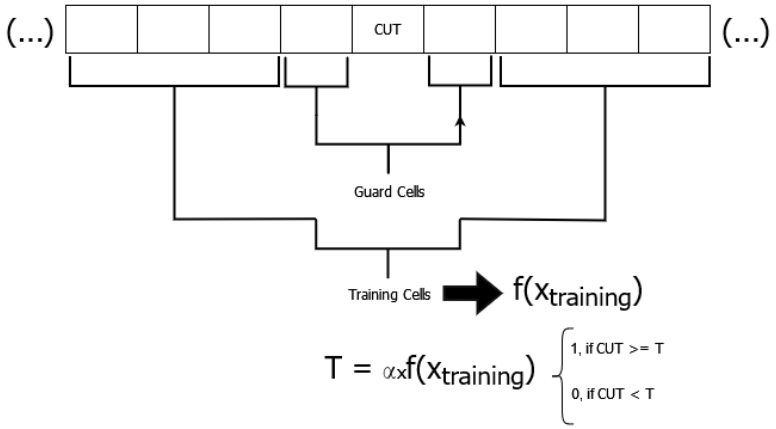


Figure 3.4: Graphical representation of the CFAR threshold algorithm

The constant false alarm rate thresholding (CFAR) is a method used to detect peaks in an array, normally applied in radar systems for target detection ([38]). It compares every value with the neighbouring values via an adaptive threshold.

Considering an array X with length N, this method loops through the array X where the value being analysed is called the cell under test (CUT). The neighbours of the CUT can be considered as guard or training cells. They can be on each side of the CUT or just on one side. Guard cells are the closest neighbouring cells that are not included in the threshold calculation, working as a buffer zone. Training cells are positioned after the guard cells, and these are responsible for the threshold calculation. The possible CUTs in an array are inside the interval {Nt+Ng+1,...,N - Nt+Ng-1}, where Nt and Ng are the length of the training and guard cells, respectively. This means that the higher the length of the training cells and guard cells, the fewer CUTs are in the array.

The Eq. 3.10 demonstrates the calculation of the threshold T.

$$T = \alpha f(x); \tag{3.10}$$

where α is the false alarm rate which is a constant value in the interval $]0,1[$, and *f(x)* is a function that involves the values of the training cells. After the calculation of the threshold, it is done a comparison between the threshold and the CUT and if the CUT is higher than the threshold, that cell is not ignored.

In this dissertation, a variation of CFAR thresholding was implemented for acoustic images. It is dominated by the smallest of cell-averaging constant false alarm rate (SOCA-CFAR) and it filters the matrix using two conditions.

$$I_{i,j} > T_{absolute} \tag{3.11}$$

$$I_{i,j} > \alpha \cdot \frac{\sum_{n=i-Ng-Nt}^{i-Ng} I_{n,j}}{Nt} \tag{3.12}$$

The Eq. 3.11 is an absolute threshold ($T_{absolute}$) for each row of the matrix, where it is a fraction of the maximum intensity value in that row. The Eq. 3.12 refers to the typical function used in CFAR algorithms. In this case, CUT must be higher than the cell averaging of the training cells multiplied by the constant false alarm α. The training cells involved in Eq. 3.12 are lagging cells, which means it is only considered the cells behind the CUT. This is done due to knowing that in an acoustic image if a certain CUT has a higher intensity than the average values of previous cells, it has a great probability of being an obstacle. This way, the process is faster to filter the values and outputs filtered images with less noise.

3.2 Map Estimation

The main objective when it comes to building a map is to represent the empty and occupied spaces accurately. In this study, the mapping representation technique used to construct and visualize the environment is the Octomaps based on [19]. The environment is divided by voxels characterized by a probability of occupancy. The probability of occupancy is the chance of the voxel containing an obstacle. Depending on their probability, the voxels can be considered occupied, empty or unknown.

The results coming from the acoustic image thresholding algorithm are not reliable to apply directly to an Octomap and update the probability occupancy of the voxels. This is due to the sonar sensing noisy measurements after detecting an object. Therefore, the feature extraction is applied to the intensity matrix resulting from the acoustic image thresholding algorithm. The algorithm filters most of the noisy measurements present in the input and defines the obstacles detected in a full scan. The output of the feature extraction are points that define the occupied volume in the space.

Having the points that define the occupied volume, is done an algorithm that defines the empty space to be able to update the map with the information about the occupied and empty volumes.

3.2.1 Range to the First Feature

The range to the first feature is a feature extractor that finds the closest point resulting from the acoustic image thresholding algorithm in each step angle. This algorithm aims to be able to distinguish the empty, occupied and unknown volumes in the space. All the points with the same distance as the closest point inside the sonar beam are considered as occupied, and after that everything is considered unknown. This way, the algorithm assures the area until a certain closest point is accurately empty. This process follows similarly to what the authors in [14] and [15] developed. Eq. 3.13 defines the output matrix from the thresholding algorithm.

$$I_{i,j} = \begin{bmatrix} I_{1,1} & \dots & I_{1,m} \\ \dots & & \dots \\ I_{n,1} & \dots & I_{n,m} \end{bmatrix} \tag{3.13}$$

where i represents each step angle, j represents each bin, n represents the total number of angles in a cycle and m is the total number of measurement bins. In each step angle, the row of the matrix will always have consecutive 0 values until a measurement that is different than 0. This first measurement is the first feature detected at that step angle due to being the first value that was not filtered by the image thresholding algorithm. This process is done for all step angles, extracting the corresponded j position of the first measurement in each angle i. This way, their polar coordinates (r,ϕ) can be determined.

$$\begin{aligned} r_{closest} &= [r_1, \dots, r_j, \dots, r_m],\ where\ r_j = j \cdot step\_distance \\ \phi_{closest} &= [\phi_1, \dots, \phi_i, \dots, \phi_n],\ where\ \phi_i = i \cdot step\_angle \end{aligned} \tag{3.14}$$

The polar coordinates are converted to Cartesian coordinates that will be used for the mapping update.

$$\begin{aligned} x_{closest} &= x_0 + r_{closest} \cdot \cos(\phi_{closest}) \\ y_{closest} &= y_0 + r_{closest} \cdot \sin(\phi_{closest}) \end{aligned} \tag{3.15}$$

where x_0 and y_0 are coordinates of the central image point. Both coordinates are considered as 0 if the central image point is the vehicle's position.

Since this study aims to map a 3D space, it is required to represent the closest points in the 3D. However, this feature extractor algorithm has as input 2D information that comes from the sonar data. To transform this 2D data into 3D data, it is applied the sonar model considered in this dissertation.

The known data from the output is the coordinates of the closest points, ($x_{closest}, y_{closest}$). However, it is not known the $z_{closest}$ for each point. The sonar model of this dissertation is considering that every point in the sonar beam with the same range as the detected obstacle is considered as an obstacle. Therefore, it is done a process that, for each ($x_{closest}; y_{closest}$) with 0 as the Z coordinate,

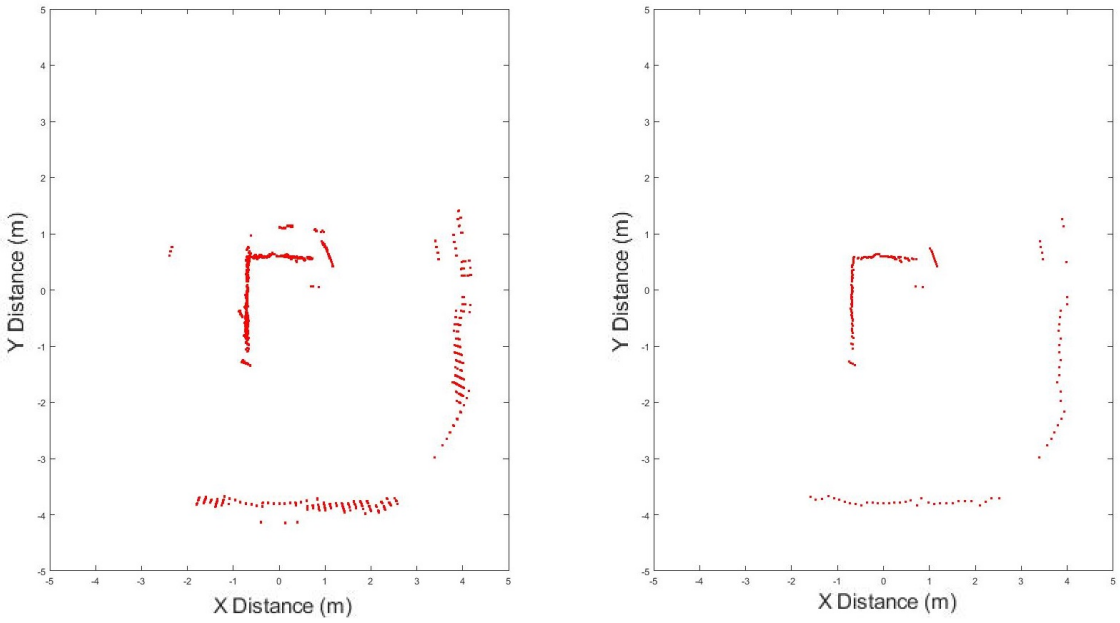


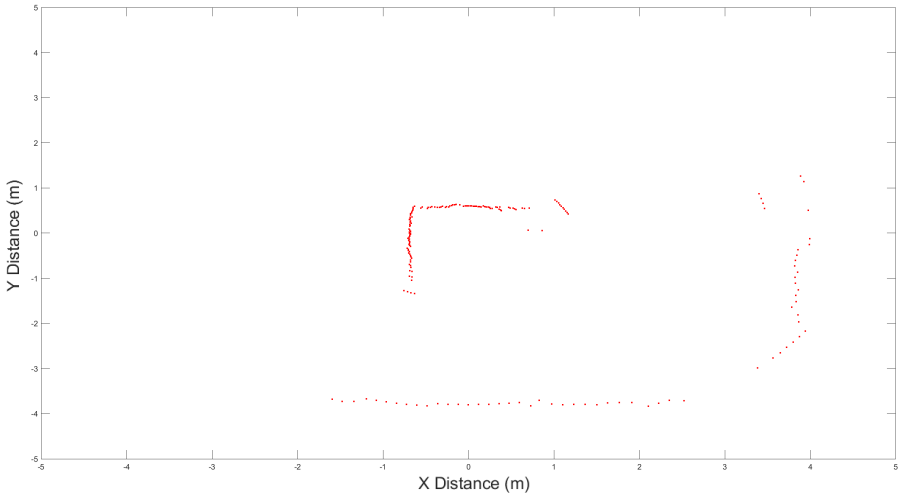
Figure 3.5: Example of the range to the first feature being applied. The left representation is the input values and right representation is the output values.

it is calculated all the points with a defined step angle, β, that are inside the sonar beam and are the same range as the closest point.

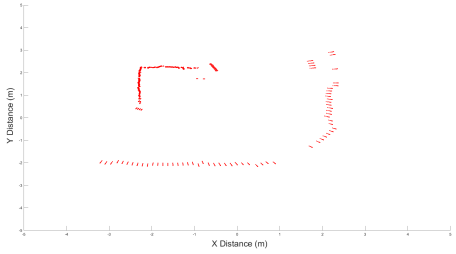
$$z_{closest} = r_{closest} \cdot \sin(\beta) \tag{3.16}$$

where β is incremented by a constant value and it is inside the interval $[-\frac{\delta_{vertical}}{2}; \frac{\delta_{vertical}}{2}]$ and $r_{closest}$ is the distance the range of the closest point.

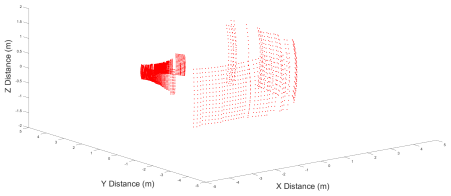
In this study, the probability attribution is direct. This means that the points representing occupied volumes are defined with a constant probability of occupancy that is high. A probability of occupancy equal to 0.9 is considered for the output of this algorithm since this value results in a good representation of the occupied volumes in the map.



(a) Plot of the 2D result



(b) Plot of the 3D result in a top view



(c) Plot of the 3D result in a side view

Figure 3.6: Example of the results obtained by the feature extractor converted to a 3D scale

3.2.2 Defining Empty Volumes

After filtering all points that correspond to the first feature detected in each step angle, it is necessary to define the empty area to be able to update the voxels of the map. It is known that, in each step angle, every point inside the sonar beam with a lower range than the closest point representing an obstacle, is in the empty space. The proposed solution is to define the empty volume of each cycle with points. To accomplish this task, it is developed an algorithm that simulates the graphical form of the sonar beam and, using a surface composed of points, checks which points are inside the sonar beam and have a lower range than the closest points.

Th Eq. 3.17 defines the 2D points that result from the first feature detected.

$$pts_{closest} = [p_1, \ldots, p_n] \tag{3.17}$$

where n is the number of points resulted in the first feature detection algorithm and $p_k = (x_k, y_k)$.

The process starts with creating a surface *surf* where its dimensions are defined to include every $pts_{closest}$ inside the limits of the surface. The *surf* is created by generating points that are separated by a constant value in each coordinate. This constant value has to be equal to or lower than the minimum length to guarantee at least one point is inside a voxel.

It is considered pt_{cut}, a point under test that is in $pts_{closest}$. It is calculated 2 distinct points,

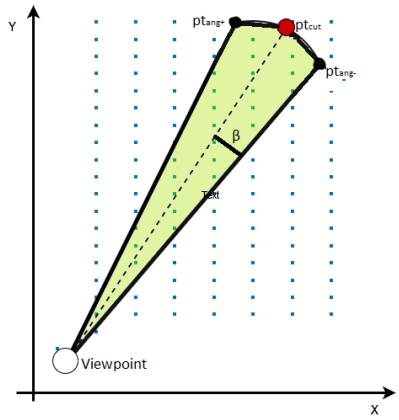


Figure 3.7: Representation of the polygon with a certain pt_{cut}. The blue points are the surface *surf* and all the points inside the polygon represent the empty volume

pt_{+ang} and pt_{-ang}, that are in the same range as the pt_{cut}, however, they differ in their angle, adding and subtracting half the angle that the sonar beam has in the horizontal plane ($\beta = \frac{\delta_{horizontal}}{2}$, where $\delta_{horizontal}$ is the horizontal angle of the sonar beam).

$$\theta = \arctan 2\left(\frac{y_{cut} - y_0}{x_{cut} - x_0}\right) \tag{3.18}$$

$$dist = \sqrt{(x_{cut} - x_0)^2 + (y_{cut} - y_0)^2} \tag{3.19}$$

$$pt_{+ang} = \left(x_0 + dist \cdot \cos(\theta + \beta),\ y_0 + dist \cdot \sin(\theta + \beta),\ 0\right) \tag{3.20}$$

$$pt_{-ang} = \left(x_0 + dist \cdot \cos(\theta - \beta),\ y_0 + dist \cdot \sin(\theta - \beta),\ 0\right) \tag{3.21}$$

These calculations are done to construct a diamond-shaped polygon, where the vertices are the origin point of the cycle (x_0, y_0), pt_{cut}, pt_{+ang} and pt_{-ang}. This polygon is meant to simulate the shape of the sonar beam with a range equal to the pt_{cut}. This means that every point that is inside this polygon is considered empty.

The surface *surf* is compared with this polygon and the points that are inside this polygon are saved. This algorithm is done for each point in $pts_{closest}$. The points saved result in the set of points that define the empty area in 2D.

The set of resulting points from this process are considered as:

$$ptsFree_{2D} = [X_{free}, Y_{free}, 0] \tag{3.22}$$

where $ptsFree_{2D}$ is the array of points with X and Y coordinates and Z = 0.

These points only define the empty area in 2D. To obtain the 3D scale, it is required to find the Z coordinates for each point in $ptsFree_{2D}$ that are inside the sonar beam volume. For each

$ptsFree_{2D}$, a process is executed that saves points with the same X and Y coordinates as $ptsFree_{2D}$ and with Z coordinates inside the sonar beam. This is done by calculating the maximum value of the Z coordinate that a point can have for each $ptsFree_{2D}$.

$$ptsFree_{2D} = [X_{free}, Y_{free}, 0] \tag{3.23}$$

$$\phi = \arctan 2\left(\frac{Y_{free} - y_0}{X_{free} - x_0}\right) \tag{3.24}$$

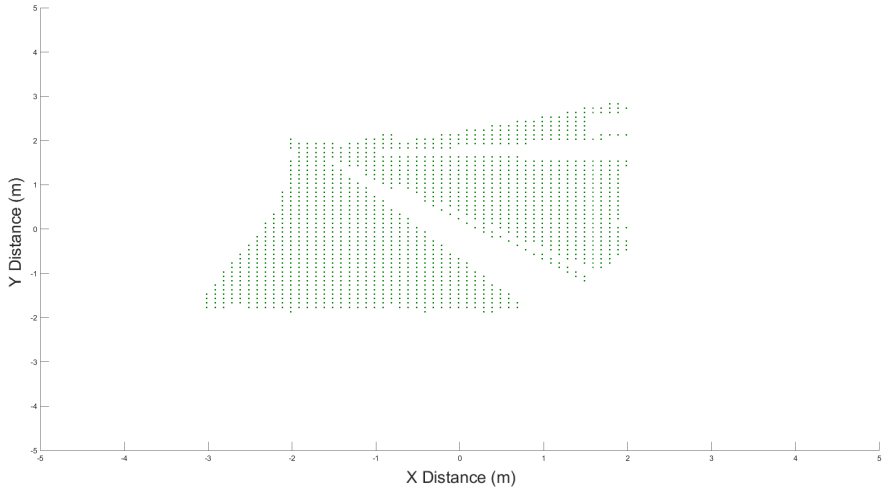
$$r_{free} = \sqrt{(X_{free} - x_0)^2 + (Y_{free} - y_0)^2}; \tag{3.25}$$

$$z_{absolute} = r_{free} \cdot \sin(\phi + \frac{\delta_{vertical}}{2}) \tag{3.26}$$

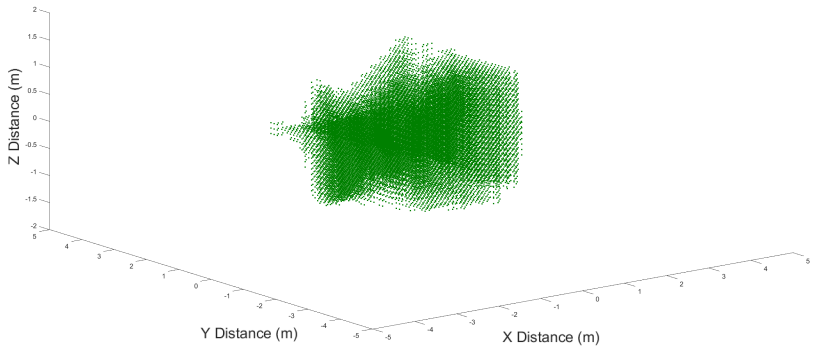
The attribution of the Z coordinates is done by starting at 0 and it is incremented with the same constant used for the surface *surf* until the Z coordinate is higher than $z_{absolute}$, while saving the points with lower Z coordinates than $z_{absolute}$. Since the points $ptsFree_{2D}$ are considered to have 0 in Z coordinate, this process saves 2 points for each incrementation. These 2 points have the same X and Y coordinates as $ptsFree_{2D}$, however, they have symmetric Z coordinates (example: if Z=0.4 is lower than $z_{absolute}$, the 2 points saved will have Z equal to 0.4 and -0.4, respectively).

The Fig. 3.8 shows the output points of this algorithm that define the empty volume which is used in the mapping update.

The output points of this algorithm are inserted into the map to update it. Since the probability attribution used in this study is defined by a constant, these points have a low probability of occupancy to represent empty volumes. It is considered that these points have a probability of occupancy of 0.25 since this value results in a good representation of the empty volumes in the map.



(a) Plot of the 3D result in a top view



(b) Plot of the 3D result in a side view

Figure 3.8: Example of the results obtained by defining empty volume with points

3.2.3 Mapping Algorithm

The Octomap algorithm implemented is similar to what Hornung et al.[19] presents. The implemented Octomap divides the 3D space into voxels with a maximum length. The data inserted into the map is saved in an Octree-type structure, where it divides the voxel into 8 sub-volumes repeatedly until the minimum length of a voxel is reached and updates the probability of the voxel.

The mapping algorithm reads the inserted points and searches the voxel that corresponds to each point. If the length of the voxel is not minimum, it is divided and if it has the minimum length, it updates the probability of occupancy of the corresponding voxel.

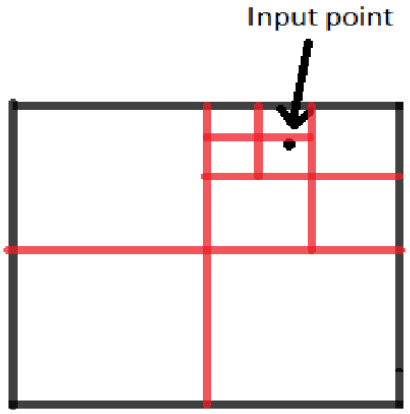


Figure 3.9: 2D representation of a voxel that is divided due to an input point

3.2.3.1 Probability Attribution and Update

When a new map is created, the probability of the space defaults to 0.5 since the environment is unknown. This probability updates with the insertion of new points to the map coming from the algorithms in the sections 3.2.1 and 3.2.2. In this dissertation, the probability update is based on log-odds notation.

The author in [19] presents the Eq. 3.27 which updates the probability of occupancy of a voxel.

$$P(n|z_{1:t}) = \left[1 + \frac{1-P(n|z_t)}{P(n|z_t)} + \frac{1-P(n|z_{1:t-1})}{P(n|z_{1:t-1})} + \frac{P(n)}{1-P(n)}\right] \tag{3.27}$$

where $P(n|z_{1:t-1})$ is a previous probability estimation and $P(n|z_t)$ is the estimation of probability of the current measurements. Applying log-odds notation and assuming a uniform prior probability, $P(n)$, leads to 0.5, the Eq. 3.27 can be simplified.

$$L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t) \tag{3.28}$$

with $L(n) = \log\left(\frac{P(n)}{1-P(n)}\right)$.

The use of log-odds notation replaces the update rule from multiplication to addition, which makes the update faster. However, the Eq. 3.29 has a characteristic that can be a problem for the system. When a voxel is considered occupied for k times, it must be considered free for k times to change its state to free space. This can be problematic since, without a limit, a voxel can be considered as occupied or free for a very long time which makes the map not very dynamic. So [19] presents a modification to the Eq. 3.29, which defines a minimum and maximum threshold for the log-odds value.

$$L(n|z_{1:t}) = max\left[min(L(n|z_{1:t-1}) + L(n|z_t), l_{max}), l_{min}\right] \tag{3.29}$$

This will ensure that the probabilities of the voxels will be bounded by l_{min} and l_{max}, adapting to new data quicker which results in a more dynamic mapping. However, this results in temporary obstacles being represented on the map for some cycles instead of ignoring them.

3.2.3.2 Clearing Inconsistencies

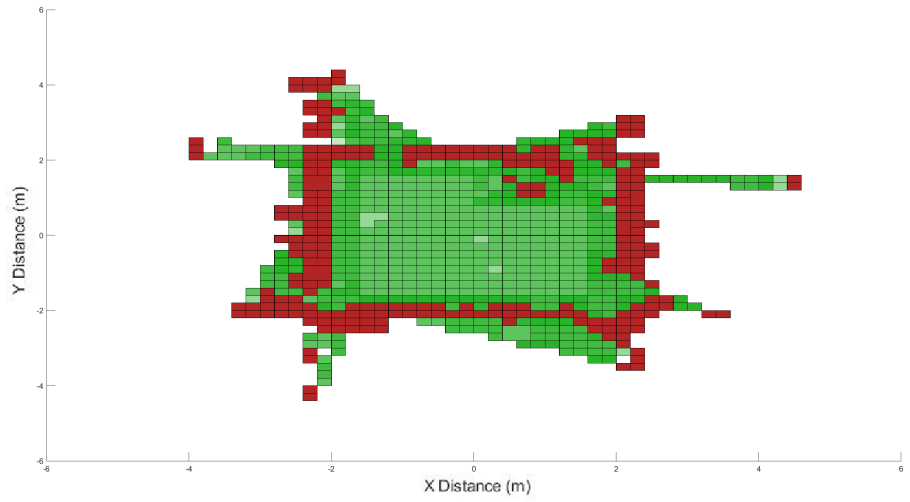


Figure 3.10: Example of an estimated map that was updated by n cycles without the algorithm that clears inconsistencies

After the map is updated with the new cycle values, the map will show some problems and incoherence that need to be addressed. Considering the situation where the map is updated with a cycle that detects an obstacle at a certain range in a certain direction. However, the next cycles detect an obstacle in the same direction but closer than the first cycle. Therefore, the first cycle detects the second reflection instead of the first reflection, which is detected in later cycles. The map is updated correctly but it results in a situation represented in the Fig. 3.10, where empty and occupied volumes are represented in front of another obstacle detected, which should not be the case.

The algorithm developed to resolve this problem is based on the same process used in section 3.2.2. Knowing the center points of all occupied voxels in 2D view, pts_{occ}, it is calculated two polygons to search for the voxels that must be cleared. It is considered $pts_{occ,i} = (x_{occ,i}, y_{occ,i})$ a point that belongs in the array pts_{occ} with an index i. The vertices of the two polygons are calculated as follows:

$$\theta = \arctan 2(\frac{y_{occ,i} - y_0}{x_{occ,i} - x_0}) \tag{3.30}$$

$$dist_{poly1} = \sqrt{(x_{occ,i} - x_0)^2 + (y_{occ,i} - y_0)^2} + offset \tag{3.31}$$

$$dist_{poly2} = \sqrt{(x_{occ,i} - x_0)^2 + (y_{occ,i} - y_0)^2} \tag{3.32}$$

$$dist_{max} = dist_{poly1} + K \tag{3.33}$$

$$\tag{3.34}$$

Polygon 1:

$$ver1_{poly1} = \big(x_0 + dist_{poly1} \cdot \cos(\theta + \gamma),\ y_0 + dist_{poly1} \cdot \sin(\theta + \gamma)\big) \tag{3.35}$$

$$ver2_{poly1} = \big(x_0 + dist_{poly1} \cdot \cos(\theta - \gamma),\ y_0 + dist_{poly1} \cdot \sin(\theta - \gamma)\big) \tag{3.36}$$

$$ver3_{poly1} = \big(x_0 + dist_{max} \cdot \cos(\theta + \gamma),\ y_0 + dist_{max} \cdot \sin(\theta + \gamma)\big) \tag{3.37}$$

$$ver4_{poly1} = \big(x_0 + dist_{max} \cdot \cos(\theta - \gamma),\ y_0 + dist_{max} \cdot \sin(\theta - \gamma)\big) \tag{3.38}$$

Polygon 2:

$$ver1_{poly2} = \big(x_0 + dist_{poly2} \cdot \cos(\theta + \gamma),\ y_0 + dist_{poly2} \cdot \sin(\theta + \gamma)\big) \tag{3.39}$$

$$ver2_{poly2} = \big(x_0 + dist_{poly2} \cdot \cos(\theta - \gamma),\ y_0 + dist_{poly2} \cdot \sin(\theta - \gamma)\big) \tag{3.40}$$

$$ver3_{poly2} = ver3_{poly1} \tag{3.41}$$

$$ver4_{poly2} = ver4_{poly1} \tag{3.42}$$

The Fig. 3.11 shows the graphical process of this algorithm. The longer polygon is to detect empty spaces and the shorter one is to detect occupied voxels. This makes the neighbouring occupied voxels that are part of the closest obstacle not being detected by the algorithm.

The detected voxels that are cells after an obstacle are forced to be 0.5 as a probability of occupancy. This algorithm corrects the problem of having empty and occupied spaces in front of obstacles, however, it creates another one. Depending on where the viewpoint is, true information can be lost due to the results of this algorithm. If in the environment, there is a corridor and the vehicle moves through the corridor and then scans one of the walls of the corridor from the outside, the information obtained inside the corridor is lost due to detecting the first wall from the outside and ignoring the rest of the voxels.

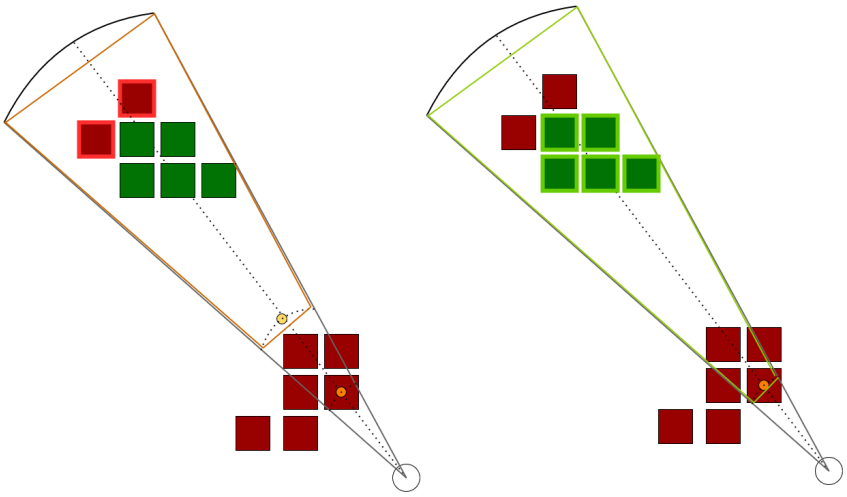


Figure 3.11: The detection of empty and occupied voxels in front of a closer obstacle

3.2.3.3 Mapping Compaction

The update algorithm always divides the voxels into 8 sub-voxels if they do not have the minimum length defined. This means that the voxels that represent empty or occupied spaces are always voxels with minimum length. This algorithm searches for the situation of having 8 occupied or empty together that can turn to a single voxel. If this situation is found, the algorithm turns the 8 sub-volumes into a single voxel with a probability of occupancy equal to the mean of the 8 probabilities of the voxels.

This way, the map is more compact and does not waste memory on saving 8 different results if they can be represented by one. This also helps in the processing speeds of the algorithms for plotting the map, since there are fewer cubic volumes to plot.

3.3 Localization

Since a map is estimated, it is now necessary to estimate the position of the vehicle within the map. This is necessary since, without the knowledge of the AUV's position, the map becomes uncertain. A system with localization implemented is capable of shifting and matching the current measurements obtained with the previous map estimation correctly.

The pose estimation developed in this study is the Particle Filter (PF). This filter is easily applied with the Octomap approach and provides a good estimation of the localization without needing various iterations. PF needs to generate a high number of particles to result in a good estimation. However, the processing time increases with the number of particles created by the algorithm. The implemented localization estimation follows the same steps that are present in [5] and [36]:

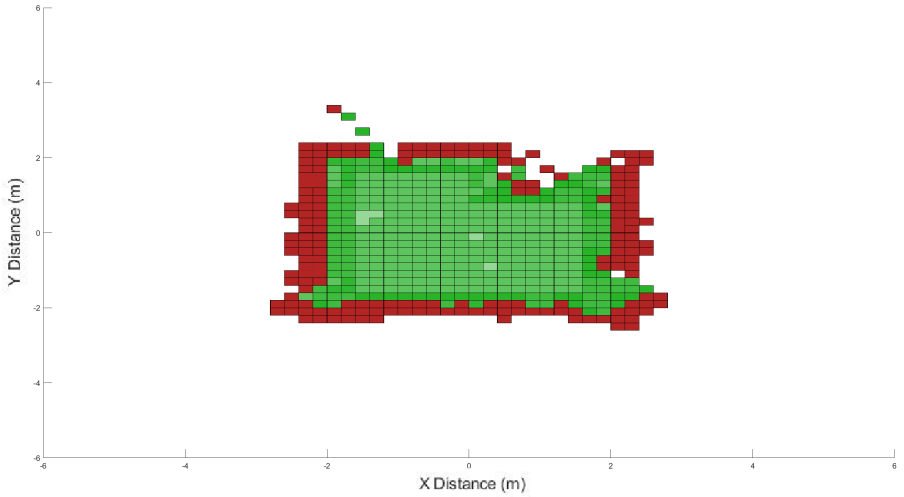


Figure 3.12: The same example in Fig. 3.10 with the algorithm that clears inconsistencies implemented

A Create a fixed number of particles in random areas inside the map and applies a weight to each particle;

B It is done a prediction using a motion model and applies this prediction to all the particles, comparing what particles moved in the same relative way as the vehicle. In this process, it is added a random motion, since the prediction of the motion model is not perfect;

C Update the weight of the particles according to the comparison done in step B;

D It is done an estimation of the position based on the weighted mean of all particles;

E It is done a re-sampling of the problem, where the less weighted particles are replaced by particles that are generated in areas where the probability of the vehicle position is higher;

F Repeat the process from step B;

The first step is to initialize the particles in the system. The PF generates a set number of initial particles randomly inside a sphere around the initial position of the vehicle. This step is executed at the beginning of the system and only executed once. All particles are defined by their position $p_k = \{x_k, y_k, z_k\}$, their orientation θ_k and their respective weight ω_k. The weight of initial particles is defined by $\frac{1}{N}$, where N is the total number of particles.

The next step is to predict the movement of the vehicle. With this prediction, the same motion is applied to the particles. This prediction is done by applying the same motion done in the last iteration. Eq. 3.43 demonstrates the calculation to obtain this motion.

$$\widehat{V}_{k-1} = \frac{\Delta\widehat{X}_{k-1}}{\Delta t_{k-1}} \tag{3.43}$$

Where $\Delta\widehat{X}_{k-1}$ is the distance travelled in the previous state and Δt_{k-1} is the time taken travelling $\Delta\widehat{X}_{k-1}$. Since the movement of a vehicle does not change drastically, we can predict the distance travelled of the current state with the previous motion.

$$\Delta\widehat{X}_k = \widehat{V}_{k-1} \times \Delta t_k \tag{3.44}$$

This prediction is applied to every particle of the filter, adding the predicted distance travelled to their position.

$$p_k = p_{k-1} + \Delta\widehat{X}_k + \delta \tag{3.45}$$

where δ is a random generated noise to the prediction, since this motion model is not perfect.

The following step is the update of weights, where each particle is classified on how close they are to the measurement done in the current state. The objective of this step is to compare the distance between the particle and the closest occupied voxel on the map with the distance between the position of the vehicle and the closest feature detect in the current state for each direction. To define this direction, it is divided the full 360º into step angles, incremented by a defined θ. In observation obtained in the current state, it is calculated the distance of the closest feature detected.

$$X_{k,i} = \sqrt{x_{k,i}^2 + y_{k,i}^2 + z_{k,i}^2} \tag{3.46}$$

Where $(x_{k,i}, y_{k,i}, z_{k,i})$ are the coordinates of the closest feature in the state k in step angle $i \times \theta$. This step is done for each particle, calculating the distance between the particle and the closest voxel in the same step angle. The weight of each particle is calculated using the normal probability density function:

$$\omega_{k,i} = \frac{1}{N\sigma\sqrt{2\pi}} e^{\frac{-(X_{k,i}-\mu_{k,i})^2}{2\sigma^2}} \tag{3.47}$$

Where $X_{k,i}$ is obtained in 3.46, $\mu_{k,i}$ is the distance between the particle and the closest voxel and σ is an acceptance threshold. Since this weight is for each step angle, the update of the weight is done as follows:

$$\omega_k = \omega_{k-1} \times \prod_{i=1}^{\frac{360}{\theta}} \omega_{k,i} \tag{3.48}$$

Since the weights resulting in Eq. 3.48 are quite low values, the next step is to normalize the weights for each particle. This normalization consists of a simple step that makes the summation

of the weight of all particles equal to 1.

$$\omega_j = \frac{\omega_j}{\sum_{n=1}^{N} \omega_n} \tag{3.49}$$

Where ω_j represents each particle. Having all the weights normalize, the estimation of the position is done by calculating the weighted mean of all particles.

$$\widehat{X}_k = \frac{\sum_{n=1}^{N} p_{k,n}\omega_{k,n}}{\sum_{n=1}^{N} \omega_{k,n}} \tag{3.50}$$

Lastly, it is necessary to do a resampling of the particles to estimate the position for the next cycle. Since the PF is normally a heavy process in terms of processing time, the systematic resampling method was chosen which is a light method and keeps the number of particles constant. The resampling implemented in this dissertation is based on the [22] method presented in the table 3.1.

Table 3.1: Pseudocode for the systematic resampling method based in [22]

Algorithm[p_{k+1},ω_{k+1}] = sysResampling(p_k,ω_k,N)
1. j=1;
2. sumW = ω_k(j)
3. u = rand()/N;
4. for i=1:N
5. while sumW <u
6. j = j+1;
7. sumW = sumW + ω_k(j);
8. endwhile
9. p_{k+1}(i) = p_k(j);
10. u = u + 1/N;
11. endfor
12. ω_{k+1} = 1/N;

Chapter 4

Experimental Setup and Results

For the development of the system, all the measures were obtained by performing various readings with an MSIS equipped in a SHAD AUV. They were taken in a water tank with a rectangular shape and its dimensions are 4.39 m by 4.58 m with 1.8 m of depth. To keep the variations of intensities low, the tank is mostly made of the same material. The data for the solution developed is 360° scans that were acquired and it was set to constant the angle resolution to 1.8°.

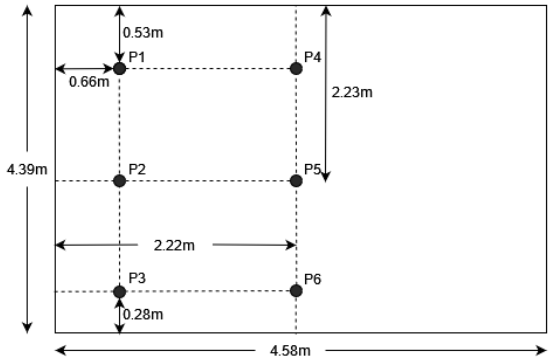


Figure 4.1: Tank dimensions and positions measurements

The datasets used for this study can be classified into two types. The first type used was static scans. These were taken from 6 distinct static positions (P1 to P6 which are represented in Fig. 4.1) in the tank, where in each position it is done various scans, varying the SONAR gain from 0.25 to 1.0 in 0.25 increments and the maximum range to 5m and 10m. This results in 48 different scans characterized by a 200x399 matrix with intensity values. The second type of dataset is a continuous reading of the tank while the AUV is travelling within the tank. The vehicle does 2 laps inside the tank in this dataset. Both datasets are used to test the different algorithms implemented in the system.

4.1 Acoustic Image Filtering

The first process implemented in the system is the acoustic image filter, where it was chosen 4 types of different thresholding algorithms. To test these 4 different algorithms, 3 different acoustic images were chosen to be filtered. These acoustic images are all in the same position, P1, and all have the same 5m range. However, they differ in their gain, using the gains 0.25, 0.5 and 1. The Fig. 4.2 represents the desired output from this algorithm using the referred acoustic image.

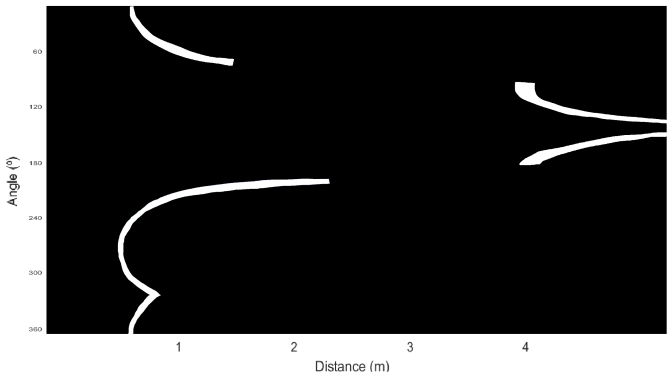
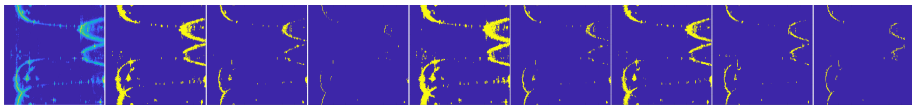
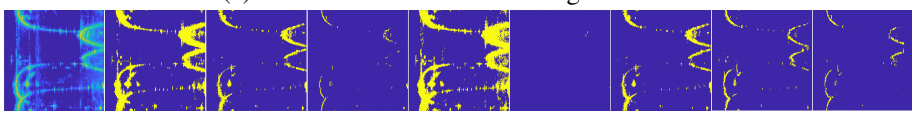


Figure 4.2: Desired output of the thresholding algorithm

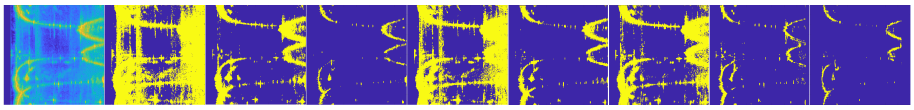
For the thresholding algorithms, different parameters were set to observe the difference in the result with the same acoustic image.



(a) Threshold results with sonar gain as 0.25



(b) Threshold results with sonar gain as 0.5

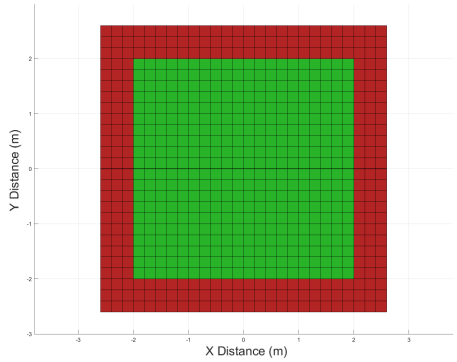


(c) Threshold results with sonar gain as 1

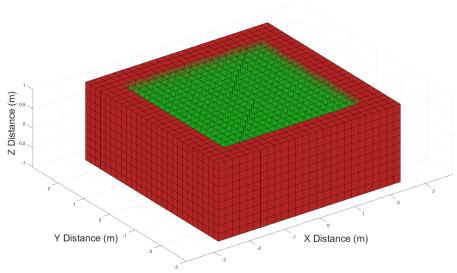
Figure 4.3: Experimental results of different acoustic image thresholding algorithms

The Fig. 4.3 is the results generated from the 4 different image thresholding algorithms implemented. From left to right, the images correspond to the following description:

- Original acoustic image;
- Simple thresholding algorithm with 30% of maximum intensity as the threshold;
- Simple thresholding algorithm with 50% of maximum intensity as the threshold;



(a) Top view



(b) 3D view

Figure 4.4: Representation of the perfect map

- Simple thresholding algorithm with 70% of maximum intensity as the threshold;
- Otsu thresholding;
- Multi-Otsu thresholding with 5 grey levels that filters the intensities of the last level;
- Multi-Otsu thresholding with 5 grey levels that filters the intensities of the last two levels;
- SOCA-CFAR with false alarm rate of 0.2, 5 guard cells, 10 training cells and absolute threshold equal to 67.5% of maximum intensity;
- SOCA-CFAR with false alarm rate of 0.2, 10 guard cells, 25 training cells and absolute threshold equal to 82.5% of maximum intensity;

When it comes to choosing an acoustic threshold algorithm, several options are reasonably good thresholding methods. In this dissertation, the SOCA-CFAR was the one being implemented due to being consistent in detecting the pretended polar representation with a few noisy measurements, despite the difference in the sonar gain. The other advantage of the SOCA-CFAR is the flexibility of adjusting the values inside the filter that fit better the characteristics of the sonar, since it is possible to adjust the number of training and guard cells, the false alarm rate and the absolute threshold of the filter.

4.2 Building a Map

In the beginning, the initial creation of the map is done. The maximum length of the voxels that initialize this map is 12.8 m and the minimum length defined is 0.2 m. To update the map, first, it updates with the free points and the occupied points update after. To test the mapping algorithm, it is used two different types of datasets: measurements when the vehicle is static and measurements when the vehicle is moving.

For the comparison of the maps results, it is generated an approximation of a perfect representation of the tank where the AUV is travelling (Fig. 4.4). This map is used to compare each

iteration and observe the matching voxels between the estimated map and the perfect representation. This comparison is done when the maps are divided only with minimum voxels to be able to match correctly. The results from the comparison between maps can be divided into 3 different types:

- Voxels are matched correctly: same voxels represent occupied or empty volumes in both maps;
- Voxels are matched incorrectly: same voxels represent opposite occupancy stats in both maps. This means that a voxel is considered occupied in the estimated map but the same is considered empty in the perfect map. The same is considered for the opposite situation;
- Voxels in the estimated map are considered as empty or occupied volumes whereas in the perfect map, they are considered unknown/outside of the tank.

4.2.1 Static

For the static situation, 6 different datasets are used to construct a map, where each dataset is taken in a different position. Since it is known the localization of each measurement within the environment, it is not needed position estimation to construct this map.

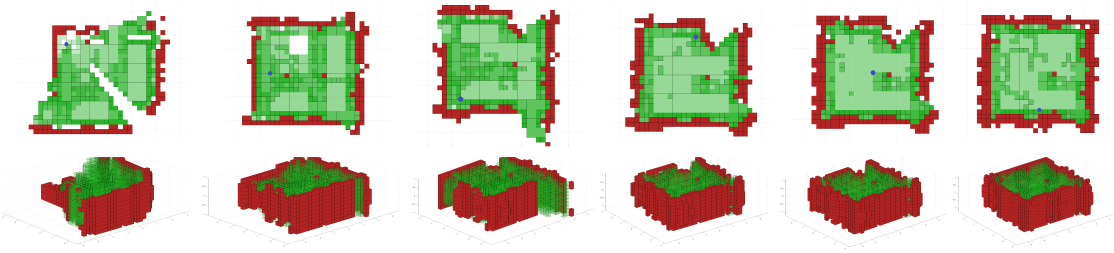


Figure 4.5: Evolution of the estimated map for each iteration with static positions (from the position P1 to P6 represented by the blue dot)

Table 4.1: Comparison between estimated map and perfect map of the occupied voxels with static positions

Iterations	Total number of occupied voxels	Total number of correctly matched voxels	Total number of wrongly matched voxels	Total number of voxels in unknown volumes
1	596	478 (80.2%)	74 (12.4%)	44 (7.4%)
2	1012	906 (89.5%)	96 (9.5%)	10 (0.9%)
3	1106	918 (83.0%)	144 (13.0%)	44 (3.9%)
4	1307	1087 (83.2%)	194 (14.8%)	26 (1.9%)
5	1460	1224 (83.8%)	218 (14.9%)	18 (1.2%)
6	1623	1424 (87.7%)	199 (12.3%)	0 (0%)

Table 4.2: Comparison between estimated map and perfect map of the empty voxels with static positions

Iterations	Total number of empty voxels	Total number of correctly matched voxels	Total number of wrongly matched voxels	Total number of voxels in unknown volumes
1	3127	2557 (81.8%)	476 (15.2%)	94 (3.0%)
2	3486	3248 (93.2%)	238 (6.8%)	0 (0%)
3	3941	3414 (86.6%)	387 (9.8%)	140 (3.6%)
4	3722	3528 (94.8%)	194 (5.2%)	0 (0%)
5	3608	3488 (96.7%)	120 (3.3%)	0 (0%)
6	3692	3642 (98.6%)	50 (1.4%)	0 (0%)

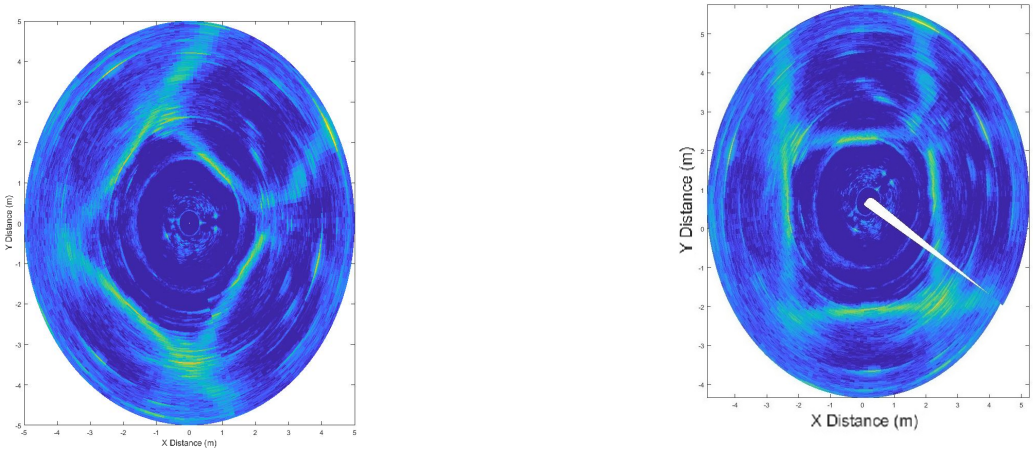
Fig. 4.5 is the evolution of the map throughout the iterations, from the position P1 to P6. Tables 4.1 and 4.2 shows the results of the comparison between the perfect map. This comparison shows the estimated map has the worst results in the first iteration, representing several free volumes where it should be occupied. However, the estimated map becomes a closer representation of the perfect map in further iterations, reducing the number of occupied and free voxels in the wrong placement and the majority of the occupied and empty voxels represent their expected state. From the table 4.1, the match of correctly occupied voxels tends to be in the 80-90% margin while the table 4.2 tends to be in the 90-99% margin except for the first iteration. This confirms the expected behaviour of the system due to the sonar model considered, showing better precision in representing empty volumes compared to occupied volumes.

4.2.2 Dynamic

In this situation, additional information was gathered. For each measurement, it was estimated the position of the vehicle by using north and east components and the heading by using azimuth angles. Since the vehicle is in motion, most of the cycles obtained will result in distorted acoustic images, that need to be undistorted. With the external information gathered provided for each measurement within a scan cycle, it is possible to define a default orientation and origin and correct the obtained acoustic images with the external information to the default position and orientation.

Since in this case, the sonar is set to 0.5 gain and 5 m range, it was done the necessary adjustments to the variables of the SOCA-CFAR to avoid reading noisy points that can disrupt the results of the map.

Since the dynamic dataset is a long continuous reading, it generates 108 iterations if divided into complete cycles. The Fig. 4.7 is the evolution of the map in the iterations 1, 40, 80 and 108 and the table 4.3 and 4.4 shows the results of the comparison between the perfect map in those iterations. The results show the same behaviour as the static situation. The first iteration is the worst estimation, but the map improves its accuracy at a certain number of iterations. However, when the vehicle is in motion, the number of wrongly matched occupied voxels shown in the table 4.3 is higher than in the static situation (increases to the 10-20% range) due to the undistortion



(a) Distorted image

(b) Undistorted image

Figure 4.6: Example of a distorted image (left) and an undistorted image (right)

process not being perfect since the external information gathered is an approximation of the real values. The table 4.4 results in similar values to the static situation due to the high certainty of the sonar model when comes to empty volumes. Overall, it is a satisfying estimation of a map and it is clear that the vehicle is inside a box-shaped structure.

4.3 Localization

To test the localization, it is performed 2 different tests. These two different tests involved locating the vehicle using the perfect map and the estimated map while the vehicle is moving in the tank. The particle filter used is initialized with 10000 particles around the initial position of the AUV. To predict the movement, the estimated motion to apply in the current iteration uses as input the north and east components and the heading that is provided in the dynamic movement.

The tables 4.5 and 4.6 show the errors in localization for the position of the vehicle in both maps. These values in the error of localization are the expected ones, being between 0.05 to 0.50 m of error for each coordinate. Both situations have similar errors which can be unexpected since one of the tests is with a perfect map representation. The reason for this happening is the observation used in the PF is the result of the feature extractor in the system, which includes some noise and temporary readings closer to the vehicle. In this case, the comparison of distances between the observation done and the map suffers significantly. Overall, the estimation of the localization generates acceptable results on where the vehicle is on the tank. The Fig. 4.8 is a plot of the ground truth trajectory done by the AUV and the two estimated trajectories by the PF.

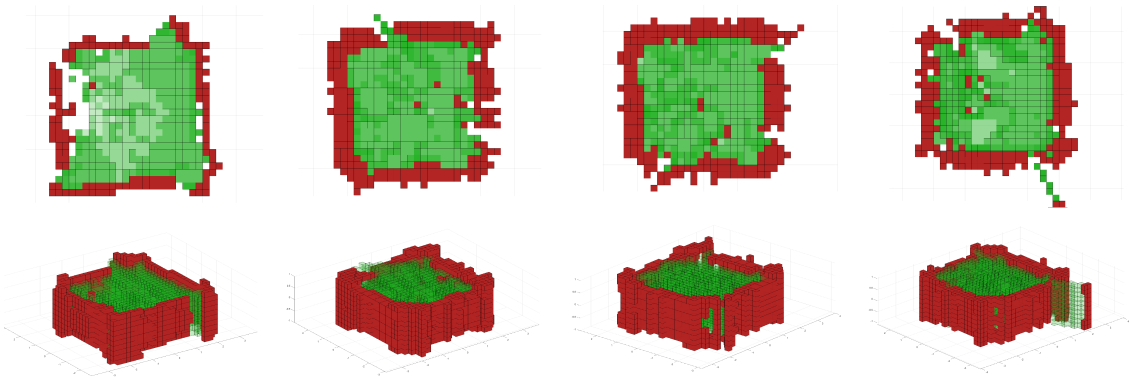


Figure 4.7: Evolution of the estimated map in 1, 40, 80, 108 iterations with vehicle in motion

4.4 Summary

This chapter, it was exposed the results obtained by the processes implemented in the system. The acoustic thresholding algorithm, the SOCA-CFAR combined with the range to the first algorithm resulted in good measurements for the mapping update, passing only a few noisy measurements. SOCA-CFAR demonstrated the advantage of being adjustable to the received values if needed. Furthermore, in the situation of maintaining the same values, the detection of the desired intensities of the acoustic image was constant with the sonar gain varying. The map update generated a good representation of the tank when the vehicle was static or travelling. The comparison to a perfect map is satisfactory since in every iteration the voxels that matched correctly in terms of occupancy are always significantly higher than the voxels that are considered wrong. For the localization, the error of the estimation averaged is the expected result since the mapping representation and the observed environment in a cycle have some differences in where features are.

Table 4.3: Comparison between estimated map and perfect map of the occupied voxels with the AUV in motion

Iterations	Total number of occupied voxels	Total number of correctly matched voxels	Total number of wrongly matched voxels	Total number of voxels in unknown volumes
1	847	699 (82.5%)	120 (14.2%)	28 (3.3%)
40	1487	1217 (81.8%)	260 (17.5%)	10 (0.7%)
80	1423	1100 (77.3%)	299 (21.0%)	24 (1.7%)
108	1309	1039 (79.4%)	244 (18.6%)	26 (2.0%)

Table 4.4: Comparison between estimated map and perfect map of the empty voxels with the AUV in motion

Iterations	Total number of empty voxels	Total number of correctly matched voxels	Total number of wrongly matched voxels	Total number of voxels in unknown volumes
1	1932	1788 (92.5%)	136 (7.0%)	8 (0.4%)
40	2400	2356 (98.2%)	44 (1.8%)	0 (0%)
80	2324	2300 (99.0%)	24 (1.0%)	0 (0%)
108	2499	2403 (96.2%)	56 (2.2%)	40 (1.6%)

Table 4.5: Errors in position of some iterations with the estimated map

Iterations	Errors [x,y,z] (m)
3	[0.09; 0.22; 0.06]
40	[0.01; 0.24; 0.24]
80	[0.06; 0.13; 0.31]
108	[0.11; 0.15; 0.25]

Table 4.6: Errors in position of some iterations with the perfect map

Iterations	Errors [x,y,z] (m)
3	[0.08; 0.21; 0.08]
40	[0.04; 0.30; 0.16]
80	[0.07; 0.23; 0.04]
108	[0.40; 0.09; 0.03]

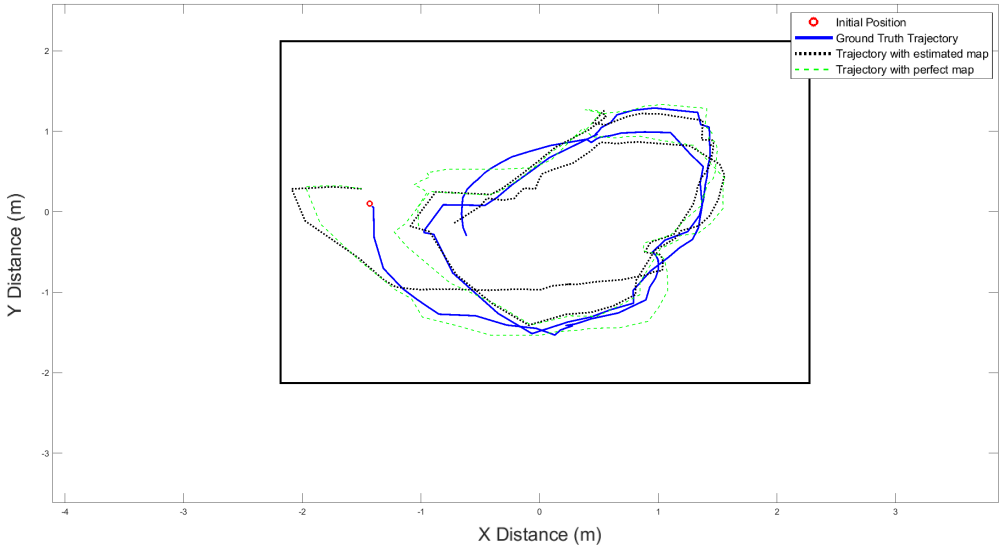


Figure 4.8: Plot of the trajectory done by the AUV and the two estimated trajectories

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In summary, this dissertation proposed a system for an autonomous map building and position estimation for an underwater vehicle equipped with an MSIS, utilizing a probabilistic 3D mapping based on Octomaps with a PF technique to estimate its position. The information was obtained by the MSIS where the sonar measurements are filtered by an acoustic image thresholding algorithm and a feature extraction to have a clear input for the map builder. The concept of the graphical representation of the sonar beam was used as a sonar model to expand the 2D representation of the sonar into a 3D map.

The main objective was to develop a system capable of extracting useful information gathered from the sonar and building a map autonomously while locating the AUV in the environment. This system accomplishes this main objective, constructing good representations of the environments while the AUV is moving or not and locates itself on the map with an expected error. However, this system has space to improve and changed to enhance the performance of the system and its application in a real-time situation.

5.2 Future Work

This system has several aspects that can be improved on that can lead to an interesting final product for real implementation. This system was tested by applying datasets that resulted in the same tank. This means it would be interesting to test and analyse the results in different environments to gain an idea of the aspects that can be improved on. On the same topic of different environments, in the map building, there is a process that needs to be improved or reworked. The referred process is the cleaning of voxels with a defined state (occupied or empty) that are in front of a closer obstacle. This improved tremendously the system for this dissertation, however, this is due to knowing that the environment is a box-shaped structure. If the AUV with this system implemented was in a tank divided into divisions and sub-spaces with entrances, the information gathered in one division

would be lost once the AUV travelled to another division, due to clearing the information after a closer obstacle.

Another aspect to consider in future work is to improve the algorithm to become a fully autonomous SLAM system. The prediction of the motion in the PF algorithm depends on the additional information about the previous positions and heading obtained externally, instead of using its previous estimation as a tool to predict the motion. A scan matching of consecutive scans can be implemented to make this system an independent SLAM process, since the undistortion is done by aligning the scans to a default position and orientation using the external information gathered for each sonar scan.

A final interesting aspect to lean on is the implementation of the system into a computational system and improving the processes in the system to reduce the processing time. It would be interesting in transforming the current system to an FPGA or GPU and take advantage of the parallelism, where it would be possible to perform certain tasks in parallel instead of being a linear approach. This way, the mapping and localization that works in real-time would be possible to implement.

Bibliography

[1] Josep Aulinas et al. “The SLAM problem: a survey”. In: *Artificial Intelligence Research and Development* (2008), pp. 363–371.

[2] Saifullahi Aminu Bello et al. “Deep learning on 3D point clouds”. In: *Remote Sensing* 12.11 (2020), p. 1729.

[3] Jaime Boal, Alvaro Sánchez-Miralles, and Alvaro Arranz. “Topological simultaneous localization and mapping: a survey”. In: *Robotica* 32.5 (2014), pp. 803–821.

[4] Lubin Chang, Jingshu Li, and Shengyong Chen. “Initial alignment by attitude estimation for strapdown inertial navigation systems”. In: *IEEE Transactions on Instrumentation and Measurement* 64.3 (2014), pp. 784–794.

[5] Ling Chen et al. “Towards autonomous localization and mapping of AUVs: a survey”. In: *International Journal of Intelligent Unmanned Systems* 1.2 (2013), pp. 97–120. DOI: https://doi.org/10.1108/20496421311330047.

[6] Jinwoo Choi et al. “EKF SLAM using acoustic sources for autonomous underwater vehicle equipped with two hydrophones”. In: *OCEANS 2016 MTS/IEEE Monterey*. IEEE. 2016, pp. 1–4.

[7] Mijo Čikeš, Marija Đakulović, and Ivan Petrović. “The path planning algorithms for a mobile robot based on the occupancy grid map of the environment—A comparative study”. In: *2011 XXIII International Symposium on Information, Communication and Automation Technologies*. IEEE. 2011, pp. 1–8.

[8] Silvia Silva da Costa Botelho et al. “Visual odometry and mapping for underwater autonomous vehicles”. In: *2009 6th Latin American Robotics Symposium (LARS 2009)*. IEEE. 2009, pp. 1–6.

[9] KTDS De Silva et al. “Comparative Analysis of Octomap and RTABMap for Multi-robot Disaster Site Mapping”. In: *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*. IEEE. 2018, pp. 433–438.

[10] Ryan M Eustice, Hanumant Singh, and John J Leonard. “Exactly sparse delayed-state filters for view-based SLAM”. In: *IEEE Transactions on Robotics* 22.6 (2006), pp. 1100–1114.

[11] Nathaniel Fairfield, George Kantor, and David Wettergreen. “Real-time SLAM with octree evidence grids for exploration in underwater tunnels”. In: *Journal of Field Robotics* 24.1-2 (2007), pp. 03–21.

[12] Dariush Forouher et al. “Sonar-based FastSLAM in an underwater environment using walls as features”. In: *2011 15th International Conference on Advanced Robotics (ICAR)*. IEEE. 2011, pp. 588–593.

[13] Hao Fu, Hanzhang Xue, and Ruike Ren. “Fast implementation of 3D occupancy grid for autonomous driving”. In: *2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. Vol. 2. IEEE. 2020, pp. 217–220.

[14] João P Fula, Bruno M Ferreira, and António J Oliveira. “Auv self-localization in structured environments using a scanning sonar and an extended kalman filter”. In: *OCEANS 2018 MTS/IEEE Charleston*. IEEE. 2018, pp. 1–6.

[15] João Pedro Bastos Fula. “Underwater mapping using a SONAR”. In: (2020).

[16] Emilio Garcia-Fidalgo and Alberto Ortiz. “Vision-based topological mapping and localization methods: A survey”. In: *Robotics and Autonomous Systems* 64 (2015), pp. 1–20.

[17] Carlos S Gonçalves, Bruno M Ferreira, and Aníbal C Matos. “Design and development of SHAD-a Small Hovering AUV with Differential actuation”. In: *OCEANS 2016 MTS/IEEE Monterey*. IEEE. 2016, pp. 1–4.

[18] Franco Hidalgo and Thomas Bräunl. “Review of underwater SLAM techniques”. In: *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*. IEEE. 2015, pp. 306–311.

[19] Armin Hornung et al. “OctoMap: An efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous robots* 34.3 (2013), pp. 189–206. DOI: https://doi.org/10.1007/s10514-012-9321-0.

[20] Stanisław Hożyń. “A Review of Underwater Mine Detection and Classification in Sonar Imagery”. In: *Electronics* 10.23 (2021), p. 2943.

[21] Fahad Jalal and Faizan Nasir. “Underwater navigation, localization and path planning for autonomous vehicles: A review”. In: *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*. IEEE. 2021, pp. 817–828.

[22] Piotr Kozierski, Marcin Lis, and Joanna Ziętkiewicz. “Resampling in particle filtering-comparison”. In: (2013).

[23] Mårten Lager, Elin A Topp, and Jacek Malec. “Underwater terrain navigation using standard sea charts and magnetic field maps”. In: *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE. 2017, pp. 78–84.

[24] John J Leonard and Alexander Bahr. “Autonomous underwater vehicle navigation”. In: *Springer handbook of ocean engineering* (2016), pp. 341–358.

[25] John J Leonard and Hugh F Durrant-Whyte. *Directed sonar sensing for mobile robot navigation*. Vol. 175. Springer Science & Business Media, 2012.

[26] Lars Linsen. *Point cloud representation*. Univ., Fak. für Informatik, Bibliothek Technical Report, Faculty of Computer . . ., 2001.

[27] Jorge L Martínez et al. "Mobile robot motion estimation by 2D scan matching with genetic and iterative closest point algorithms". In: *Journal of Field Robotics* 23.1 (2006), pp. 21–34.

[28] Facundo Mémoli and Guillermo Sapiro. "Comparing point clouds". In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 2004, pp. 32–40.

[29] Lars Mescheder et al. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4460–4470.

[30] Hans Moravec and Alberto Elfes. "High resolution maps from wide angle sonar". In: *Proceedings. 1985 IEEE international conference on robotics and automation*. Vol. 2. IEEE. 1985, pp. 116–121.

[31] Juan Nieto, Tim Bailey, and Eduardo Nebot. "Recursive scan-matching SLAM". In: *Robotics and Autonomous systems* 55.1 (2007), pp. 39–49.

[32] Pedro L Oliveira, Bruno M Ferreira, and Nuno A Cruz. "Experimental evaluation of segmentation algorithms for corner detection in sonar images". In: *OCEANS 2019 MTS/IEEE SEATTLE*. IEEE. 2019, pp. 1–10.

[33] Edwin B Olson. "Real-time correlative scan matching". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 4387–4393.

[34] Albert Palomer, Pere Ridao, and David Ribas. "Inspection of an underwater structure using point-cloud SLAM with an AUV and a laser scanner". In: *Journal of field robotics* 36.8 (2019), pp. 1333–1344.

[35] Narcís Palomeras, Marc Carreras, and Juan Andrade-Cetto. "Active SLAM for autonomous underwater exploration". In: *Remote Sensing* 11.23 (2019), p. 2827.

[36] Liam Paull et al. "AUV Navigation and Localization: A Review". In: *IEEE Journal of Oceanic Engineering* 39.1 (2014), pp. 131–149. DOI: https://doi.org/10.1109/JOE.2013.2278891.

[37] João Quintas, Francisco Curado Teixeira, and António Pascoal. "AUV geophysical navigation using magnetic data-The MEDUSA GN system". In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2018, pp. 1122–1130.

[38] Hermann Rohling. "Ordered statistic CFAR technique-an overview". In: *2011 12th International Radar Symposium (IRS)*. IEEE. 2011, pp. 631–638.

[39] Suresh Chandra Satapathy et al. "Multi-level image thresholding using Otsu and chaotic bat algorithm". In: *Neural Computing and Applications* 29.12 (2018), pp. 1285–1307.

[40] Brian Douglas The MathWorks Inc. *Understanding SLAM Using Pose Graph Optimization | | Autonomous Navigation, Part 3*. Youtube. 2020. URL: https://www.youtube.com/watch?v=saVZtgPyyJQ&t.

[41] Michael E West and Vassilis L Syrmos. "Navigation of an autonomous underwater vehicle (AUV) using robust SLAM". In: *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*. IEEE. 2006, pp. 1801–1806.

[42] Manuel Yguel, Olivier Aycard, and Christian Laugier. "Update policy of dense maps: Efficient algorithms and sparse representation". In: *Field and Service Robotics*. Springer. 2008, pp. 23–33.

[43] Mingxi Zhou, Ralf Bachmayer, and Brad deYoung. "Mapping for control in an underwater environment using a dynamic inverse-sonar model". In: *OCEANS 2016 MTS/IEEE Monterey*. IEEE. 2016, pp. 1–8.