FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Local Knowledge Representation with Topological Maps for Obstacle Avoidance in Maritime Scenarios

Pedro Sousa da Rocha

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: João Vasco Couto Amorim Marques

October 30, 2023

Resumo

O sector marítimo tem vindo a evoluir para a automatização nos últimos anos, com regulamentos já implementados. Esta evolução tecnológica abrange o desenvolvimento de Veículos de Superfície Autónomos (ASVs). A introdução destes veículos irá reduzir significativamente a presença e a intervenção humana nesta indústria e, consequentemente, reduzir a maioria das causas das catástrofes - o erro humano. Devido ao atual estado da tecnologia, não é possível a construção de embarcações à escala real. No entanto, está a ser feita investigação neste domínio, que tem tido uma tendência crescente nos últimos anos.

O mapeamento é um processo de identificação de características e limites no ambiente, que conduz a uma maior consciência situacional para as embarcações autónomas, permitindo uma melhor avaliação e gestão dos riscos e, consequentemente, uma navegação mais segura. Além disso, um mapa dinâmico é fundamental para evitar colisões. Os mapas topológicos são bem adequados para o ambiente marítimo devido à dualidade entre a falta de características no mar aberto e o ambiente complexo dos portos. Estes mapas apontam e identificam características marcantes do cenário e adicionam anotações para completar ainda mais a informação recolhida.

Com esta intenção em mente, esta dissertação tem como objetivo dar mais um passo na autonomia das embarcações, apresentando o desenvolvimento de um algoritmo adaptável e escalável para construir mapas topológicos. O algoritmo foi concebido para funcionar a bordo do Nautilus ASV e pode recolher dados de vários sensores no veículo.

O Nautilus recolheu dados para testar e validar o algoritmo navegando através de um porto próximo em Viana do Castelo e registando dados de sensores a bordo, incluindo LiDAR, câmaras, IMU e GPS. Adicionalmente, foi necessário adquirir mais dados do simulador 3D Gazebo.

O algoritmo de construção de mapas topológicos desenvolvido consegue produzir resultados aceitáveis em cenários simulados e reais, descrevendo com exatidão a envolvente do veículo.

Abstract

The maritime industry has been shifting towards automation in the last few years, with already implemented regulations. This technological evolution covers the development of Autonomous Surface Vehicles (ASVs). Introducing these vehicles will significantly reduce the human presence and intervention in this industry and, consequentially, reduce the majority of the catastrophes' causes - human error. Due to the current technological state, full-scale vessels are not possible. However, research is being done in this field, which has been an upward trend in recent years.

Mapping is a process of identifying features and boundaries in the environment, leading to an increased situational awareness for autonomous vessels allowing superior risk assessment and management and, thus, safer navigation. Furthermore, a dynamic map is fundamental for collision avoidance. Topological maps are well suited for the maritime environment due to the duality between the lack of features in the open sea and the complex environment of the harbors. These maps point to and identify landmarking features of the scenario and add annotations to complete the collected information further.

With this intention in mind, this dissertation aims to take a further step into vessel autonomy by presenting the development of an adaptable and scalable algorithm to build topological maps. The algorithm is designed to work onboard the Nautilus ASV and can take data from multiple sensors in the vehicle.

The Nautilus ASV collected data to test and validate the algorithm by sailing through a nearby port in Viana do Castelo and recording data from onboard sensors, including LiDAR, cameras, IMU, and GPS. Additionally, acquiring more data from the 3D simulator Gazebo was necessary.

The developed topological map-building algorithm can produce acceptable results in simulated and real scenarios, accurately describing the vehicle's surroundings.

Agradecimentos

Ao Prof. Dr. Andry Maykol Pinto, pela confiança depositada em mim, pela oportunidade que me foi dada, pela dedicação e apoio prestado em cada momento e pela partilha do intusiasmo pela inovação. Ao meu orientador Prof. Dr. João Vasco Couto Amorim Marques e ao meu co-orientador Daniel Campos pelo incansável apoio que possibilitou a realização desta dissertação. À restante equipa do CRAS, pela motivação e pela ajuda dispendida em cada momento.

Aos cogumelos que levo da faculdade para a vida, por terem sempre lá presentes em todos os momentos - bons e maus. Foram as experiências desde as festas até às longas noites de estudo que fizeram do curso algo do qual vou ter saudades.

Aos meus pais, pela oportunidade de continuar estudar e pela constante motivação ao longo do percurso académico. À minha irmã e ao resto da família pelo apoio constante, não só durante o curso mas durante toda a minha vida, e pelos valores que me ensinaram que com trabalho é possível alcansar qualquer coisa.

Pedro S. Rocha

"When something is important enough,
you do it even if the odds aren't in your favor."

Elon Musk

Contents

1 Introduction **1**
- 1.1 Context and Motivation . . . 1
- 1.2 Objectives . . . 2
- 1.3 Structure of the Document . . . 3

2 State of the Art **4**
- 2.1 Obstacle Avoidance . . . 4
  - 2.1.1 Potential Fields . . . 4
  - 2.1.2 Neural Networks . . . 5
  - 2.1.3 Vision-Based . . . 7
  - 2.1.4 Genetic Algorithm . . . 8
  - 2.1.5 Fuzzy Logic . . . 9
  - 2.1.6 Adaptive Velocity Obstacle Avoidance (AVOA) . . . 10
- 2.2 Scenario Representation . . . 13
  - 2.2.1 Grid-based Methods . . . 13
  - 2.2.2 Polygonal Maps . . . 14
  - 2.2.3 Voronoi Diagrams . . . 15
  - 2.2.4 Point Clouds and Voxels . . . 16
  - 2.2.5 Topological Maps . . . 17
- 2.3 Critical Review . . . 19

3 A Semantic Map Representation for Obstacle Avoidance **20**
- 3.1 Introduction . . . 20
- 3.2 Detectors . . . 22
  - 3.2.1 Vessel and Buoy Detection . . . 22
  - 3.2.2 Dock Detection . . . 23
  - 3.2.3 Detector Standardization . . . 24
- 3.3 Scenario Representation . . . 27
  - 3.3.1 Odometry . . . 27
  - 3.3.2 Features and Landmarks . . . 28
  - 3.3.3 Visualization . . . 30

4 Results **32**
- 4.1 Scene perception and understanding . . . 32
  - 4.1.1 Dock detector . . . 32
  - 4.1.2 Vessel detector . . . 34
- 4.2 Scenario representation . . . 36
  - 4.2.1 Manually controlled data inputs . . . 37

4.2.2 Simple scenario in a simulated world 37
4.2.3 Complex scenario with dynamic objects in a simulated world 38
4.2.4 Real scenario in the ATLANTIS test center 41
4.2.5 Performance Analysis 43

5 Conclusions and future work **44**

References **46**

List of Figures

1.1 Causes of passenger vessel accidents (left) and fatalities (right) [1] 2
1.2 INESC TEC's SENSE ASV [2] 3

2.1 Visual representation of the concept of the potential field. The attractive potential without obstacles (left), the repulsive potential representing the obstacles (center), and the final potential field (right). 5
2.2 Path resulting from a wandering algorithm using obstacle avoidance based on deep reinforcement learning [3]. 6
2.3 Comparision between a neural network and human inputs [4]. 7
2.4 Example of visual odometry [5] 8
2.5 Representation of the protective zone concept [6] 11
2.6 Possible velocity vectors constrained by vessel's kinematics (left) of a single static obstacle and (right) with a static and a dynamic obstacle [6] 11
2.7 Example of a robot and an obstacle in a 2D occupancy grid [7] 13
2.8 Approximation of a noisy 2D LiDAR measurement into two lines 14
2.9 Voronoi diagram example [8] 15
2.10 Voxel representation in of the two original objects (a) in different resolutions [9] . 17
2.11 Two topological maps with different obstacles representation 18

3.2 Proposed architecture scheme for semantic mapping and obstacle avoidance using multiple detectors. 22
3.3 Average precision comparison between different YOLO versions [10]. 23
3.4 Distribution of the occurrence of each class in the entire dataset. 24
3.5 Example of successful detection of a boat. 25
3.6 SENSE ASV and a dock in the Gazebo simulator (left) and successful dock detection visual data representation (right). 26
3.7 Example of a reference frame transformation from the robot frame into the odometry frame. 26
3.8 Architecture used to implement the topological map. 27
3.9 Behavior of the algorithm when observing a new landmark. 29
3.10 Example of each possible marker used for the visual scenario representation. The green arrow represents an odometry node, the shares represent landmark nodes (turquoise if stationary or purple if moving), the white ellipsis represents its associated node covariance and the lines connect the nodes (yellow connects odometry nodes, orange connects odometry nodes with stationary landmark nodes and red connects odometry nodes with moving landmarks). 31

4.1 Custom trained vessel detector's confusion matrix. 34
4.2 Custom trained vessel detector's F1 curve. 35

4.3 Custom trained vessel detector's precision-recall curve. 36
4.4 Map's visual representation resulting from manually feeding odometry points and features into the map builder algorithm. Vertices two and four are stationary landmarks, vertex six is a moving obstacle, and vertices zero, one, three, five, seven, eight, and nine are odometry nodes. 38
4.5 Simulated environment test in a simple scenario - three docks are positioned in a triangle configuration, and the ASV navigates with an S-shape trajectory between the U-shaped and T-shaped docks. 39
4.6 Complex scenario with dynamic obstacles test - the static objects are the same as for the previous test. The dynamic object consists of a large boat that crosses the robot's path, including one of the docks at the beginning of the test. 40
4.7 Real scenario test - six vessels are docked in the ATLANTIS test center. The ASV navigates around the central dock, represented by the yellow ellipsis. 42
4.8 Data collected by the logger for posterior analysis. 43

List of Tables

4.1 Confusion matrix for a binary classification task. 33
4.2 Accuracy of the detection stage. 33
4.3 Precision and accuracy of the detection stage. 34
4.4 Accuracy of the detection stage. 35
4.5 Precision and accuracy of the classification stage. 36

Acronyms and Symbols

ASV	Autonomous Surface Vehicle
USV	Unmanned Surface Vehicle
IMO	International Maritime Organization
OECD	Organisation for Economic Co-operation and Development
MSC	IMO's Safety Committee
3D	Three-dimensional
2D	Two-dimensional
2.5D	Two-and-a-half-dimensional
LiDAR	Light Detection and Ranging
IMU	Inertial Measuring Unit
GPS	Global Position System
CCD	Charged Coupled Devices
CMOS	Complementary Metal Oxide Semiconductors
CNN	Convolutional Neural Network
RNN	Recurring Neural Network
TEB	Time Elastic Band
ROS	Robot Operating System
PCL	Point Cloud Library
UAV	Unmanned Aerial Vehicle
AVOA	Adaptive Velocity Obstacle Avoidance
SLAM	Simultaneous Mapping And Location
EKF	Extended Kalman Filter
UTM	Universal Transverse Mercator

Chapter 1

Introduction

1.1 Context and Motivation

The global economy depends on the trading of products worldwide. According to the International Maritime Organization (IMO), "*shipping is the most efficient and cost-effective method of international transportation for most goods*"[1], making up nearly 90% of the total international transportation. The Organisation for Economic Co-operation and Development (OECD) predicts that the freight demand will increase and the maritime trade volumes will triple by 2050[2].

This high number of dislocations reflects a large number of accidents, which cause a negative impact on the environment and economy and can lead to human casualties. While an unmanned ship is ideally impossible to sink under ordinary conditions, a manned craft is always prone to human error, causing it to fail and, in some cases, plunge. Grounding, collisions, negligence, and human error are among the most common reasons why ships sink[3]. Figure 1.1 illustrates the distributions of passenger vessels' accident causes.

To address these problems, Autonomous Surface Vehicles (ASVs) are being introduced, and IMO's Strategic Plan (2018-2023) has a key Strategic Direction to "*integrate new and advancing technologies in the regulatory framework*." In 2017, IMO's Safety Committee (MSC) agreed to include the issue of marine autonomous surface ships on its agenda[4]. An ASV is an unmanned vessel that can perform different tasks such as commercial transportation, offshore energy generation inspection and maintenance, environment monitoring, exploration, mapping, and military applications. These vehicles were initially conceived to decrease significantly the risk for humans on the previously listed activities. Unmanned Surface Vehicles (USVs) can be remotely controlled by a human operator located on land or a nearby ship, or they can use a set of sensors and the onboard processing unit to be autonomous. Figure 1.2 shows an example of an ASV.

[1] IMO, Introduction to IMO, 2022. https://www.imo.org/en/About/Pages/Default.aspx

[2] OECD, Ocean shipping and shipbuilding, 2022. https://www.oecd.org/ocean/topics/ocean-shipping/

[3] Marine Insight, Why Ships Sink – 10 Major Reasons, 2022. https://www.marineinsight.com/naval-architecture/why-ships-sink-10-major-reasons

[4] IMO, Autonomous shipping, 2022. https://www.imo.org/en/MediaCentre/HotTopics/Pages/Autonomous-shipping.aspx

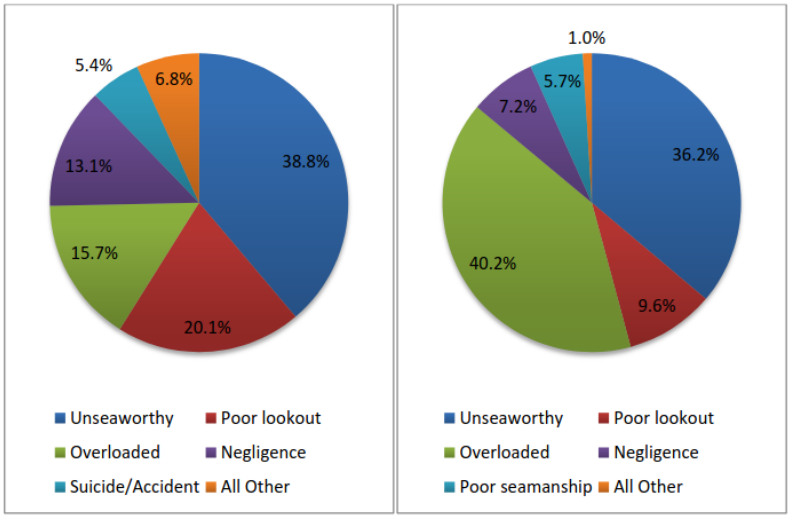


Figure 1.1: Causes of passenger vessel accidents (left) and fatalities (right) [1]

To be fully autonomous, ASVs require an obstacle-avoiding system to prevent collisions with other vessels, buoys, harbors, and maritime structures they may encounter. For that task, a representation of the environment is mandatory to describe the location of the obstacles ASVs need to evade. Although a metric map can represent the environment entirely, it does not characterize the surrounding obstacles. Accurately characterizing each object in the scene contributes positively to calculating the best possible maneuver to avoid these obstacles.

Therefore, this dissertation proposes a topological map-based semantic representation to improve the traditional metric maps' representation. To achieve this representation, detecting and characterizing vessels, docks, buoys, and other objects according to their type, pose, velocity, and size is necessary. This object characterization is important in the decision-making process of evading each type of obstacle, as each object may require different maneuvers.

1.2 Objectives

This dissertation aims to provide an ASV with the capability of mapping its surroundings while avoiding all obstacles autonomously. The ASV is equipped with cameras, a 3D (three-dimensional) LiDAR, Global Position System (GPS), IMU, and other sensors, allowing the capability to collect the required data to assemble the map and its representation and collision-free navigation.

In this regard, this dissertation aims to:

- Integrate several deep learning detectors using multi-modal data, namely LiDAR and cameras. This allows the detection of objects such as other vessels, docks, and other uncategorized obstacles that may appear;
- Characterize the detected objects regarding their dynamics, position, class, and size;

Figure 1.2: INESC TEC's SENSE ASV [2]

- Develop a scene representation methodology for obstacle avoidance capable of associating heterogeneous data into a single map;
- Validate the methodology using a simulated environment and in the ATLANTIS test platform [11] using a real ASV.

1.3 Structure of the Document

Besides the introduction, this dissertation contains four other chapters:

- Chapter 2 describes the state-of-the-art, identifying several methods to solve previously described problems. There are three sections in this chapter. The first is about obstacle avoidance, the second is about detection and scenario representation, and the last section critically reviews the most relevant works analyzed. It is also an exposition of the fundamentals of perception and obstacle avoidance.
- Chapter 3 presents a detailed explanation of each individual component of the proposed methodology.
- Chapter 4 contains the results and their respective analysis of the conducted experiments.
- Chapter 5 summarizes all drawn conclusions, as well as the future work opportunities derived from this work.

Chapter 2

State of the Art

2.1 Obstacle Avoidance

In mobile robotics, obstacle avoidance is fundamental to safe and efficient navigation. Several methods were developed to handle that challenge, and in this context, four main approaches arise sensor-based, control-based, and hybrid methods.

The first approach, sensor-based obstacle avoidance, relies on sensors, such as LiDAR or cameras, to perceive the surrounding environment [12]. With sensors, the robot can collect data and navigate with a low computational power algorithm. However, the sensors' precision and range can limit the quality and safety of navigation and obstacle avoidance.

The control-based obstacle avoidance relies on predicting the future robot state using known environment properties [13]. By modeling the robot's dynamics and their interaction with the environment, it is possible to plan obstacle avoidance maneuvers. This method requires a deep understanding of the dynamics and interactions mentioned, making achieving precision challenging.

Hybrid methods combine the advantages of the previously mentioned methods to obtain better and more reliable results. By integrating sensor-based perception and predictive control, these methods deliver improved capabilities in obstacle avoidance. These solutions are often more robust in complex scenarios.

A survey conducted by Aggrey Shitsukane [14] compares different algorithms and their advantages and limitations. The choice of the method depends on factors such as available sensors, computational resources, environment complexity, and desired performance. Additional research in this area aims to improve the robustness, efficiency, and adaptability of obstacle avoidance algorithms for mobile robotics.

2.1.1 Potential Fields

The method of potential fields for robot obstacle avoidance was proposed for the first time in the 1980s [15]. These methods were deeply studied and applied to mobile robotics due to their intuitive approach and efficiency in avoiding collisions and guiding the robot to its objective.

The fundamental idea behind the potential fields method is to create a virtual force field around the robot, which acts as a "force" that pushes it away from the obstacles and toward the desired objective. This approach is inspired by the physics concept, where the particles are influenced by forces that direct them into a minimum of energy. It has been broadly used due to its conceptual simplicity and capability to generate smooth and continuous trajectories for the robot.

In the context of potential fields, the field's force and direction are determined by the distance and location of the obstacles in the environment. Usually, obstacles are represented by "peaks" of repulsive fields, while the target destination is represented as an attractive field "valley." Figure 2.1 visually represents this concept. This way, the robot is guided by the gradient of the potential field, following a trajectory toward the objective and avoiding obstacles simultaneously.

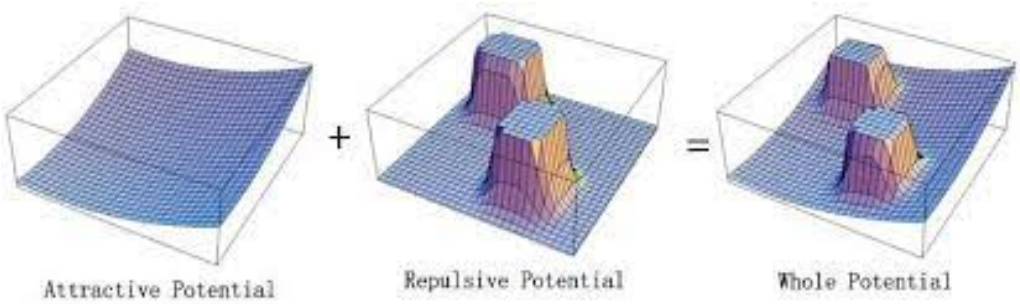


Figure 2.1: Visual representation of the concept of the potential field. The attractive potential without obstacles (left), the repulsive potential representing the obstacles (center), and the final potential field (right).

One of the advantages of the potential field method is its implementation simplicity. The idea is to create a virtual potential field that algorithms can easily understand and implement. Furthermore, these methods can handle dynamic environments, given that the force calculation can be updated in real-time as the robots move and the obstacles are modified.

However, the potential field method has some limitations. One of the most common challenges is the possibility of the robot getting stuck in a local minimum or oscillating around obstacles. This occurs when the repulsive forces overcome the attractive force, making the robot not follow an ideal path. Furthermore, multiple obstacles close to each other (e.g., a narrow door) may lead to unfavorable competition between repulsive and attractive forces.

To overcome these challenges, several strategies were proposed in the literature [16]. A standard solution involves the use of stochastic approaches, such as ant colony optimization[17], simulated annealing [18], or Monte Carlo simulation [19] to allow the robot to escape local minima and keep following the potential field. These techniques introduce random elements in the robot's trajectory, allowing it to explore different paths and avoid getting stuck in local minima.

2.1.2 Neural Networks

Neural networks play a crucial role in the field of autonomous robotics, especially in the domain of navigation and obstacle avoidance. These techniques have been extensively researched due to

their capability to handle complex and unpredictable environments.

A commonly adopted approach involves the integration of neural networks in obstacle avoidance systems. These systems generally consist of sensors, such as cameras, LiDARs, and SONARs, which provide information about the surrounding environment. Collected data is processed by neural networks, which can be designed with different architectures, such as convolutional neural networks (CNNs) or recurring neural networks (RNNs). Figure 2.2 shows an example of an RNN applied to obstacle avoidance.

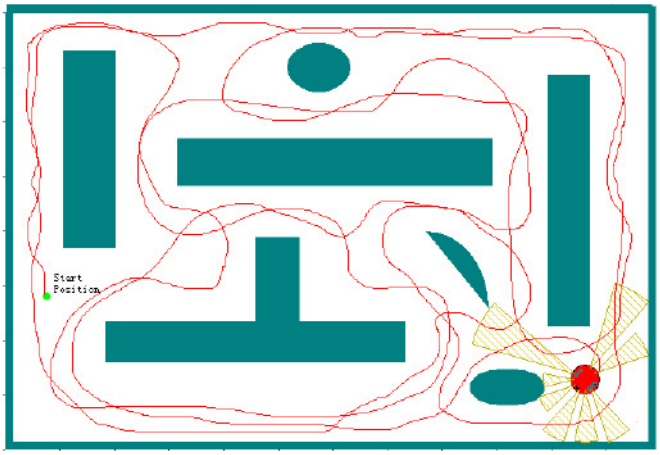


Figure 2.2: Path resulting from a wandering algorithm using obstacle avoidance based on deep reinforcement learning [3].

Another noticeable advance in neural networks is the ability to create systems that imitate and match human inputs [20]. Through these techniques of deep learning and sophisticated computational modeling, neural networks have been able to learn and recognize sensor data similarly to humans. This ability to match human inputs is fundamental in autonomous obstacle avoidance. Figure 2.3 compares human and machine learning inputs of a robot navigating a room. The capability of neural networks to approach human cognitive processing has been one of the main boosters in artificial intelligence research, allowing AI systems to be more adaptable, intuitive, and efficient in interacting with users [21].

One of the main advantages of neural networks in obstacle avoidance is their capability to learn ad adapt from a data set. During the training phase, the neural network is exposed to various scenarios and configurations of obstacles, allowing it to capture patterns and complex relations between input data and desired output actions. This learning ability allows the neural network to generalize knowledge and make suitable decisions in never-seen situations.

Another important aspect is the use of reinforcement learning techniques in obstacle avoidance. In this context, neural networks are trained through continuous interactions with the environment, receiving feedback as rewards or penalties based on their performance.

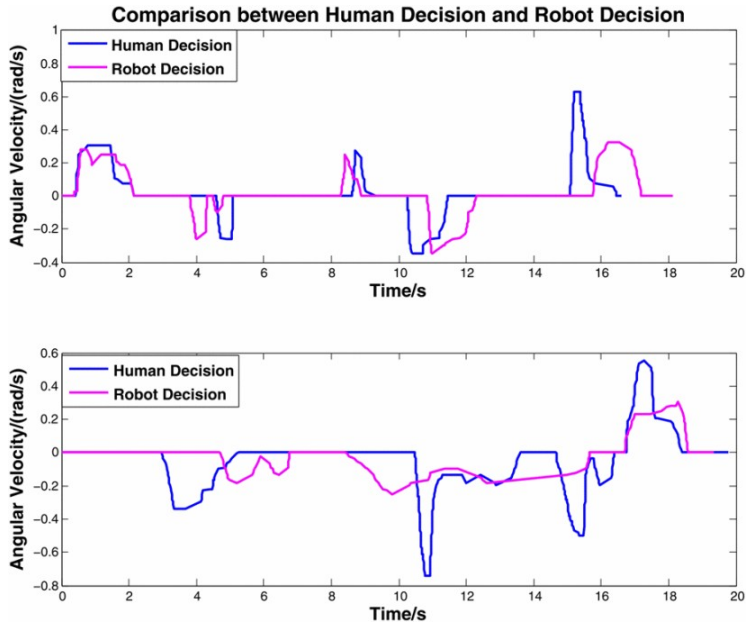


Figure 2.3: Comparision between a neural network and human inputs [4].

2.1.3 Vision-Based

Visual odometry is a robot's movement estimation method using visual information. It involves using a camera to capture images of the scenario as the robot moves and using image processing techniques to estimate the robot's pose (position and orientation). It is a popular approach in robotics and autonomous navigation because it allows robots to obtain information about their movement relative to their surrounding environment without depending exclusively on external sensors, such as GPS or odometry-based encoders [22].

Visual odometry plays a critical role in several applications, especially in Unmanned Aerial Vehicles (UAVs) or any other vehicle where the mass of the sensors matters. It allows a precise estimate of the robot's movement, allowing it to localize itself in the environment and navigate autonomously. Furthermore, visual odometry can be a viable alternative in places where other sensors can be limited, such as areas with low GPS signals or uneven terrain where odometry-based encoders can be imprecise [23, 24].

There are several approaches to visual odometry, including feature-based or direct methods. Feature-based methods involve identifying distinct points in the image and tracking their movement over time to estimate the robot's pose [25]. An example of this process is shown in image 2.4. On the contrary, direct methods involve the direct estimate of the movement from the images without needing identifiable features [26]. These approaches have their respective advantages and challenges, and the choice depends on the context application or the performance requirements.

Furthermore, visual odometry has several advantages and relevant challenges that must be considered. One of the main advantages of this method is the capability of obtaining detailed information about the environment in real-time, allowing for a rich visual perception of the robot.

Figure 2.4: Example of visual odometry [5]

This capability allows for a greater comprehension of the surrounding scenario, favoring navigation and interaction with the environment. However, implementing visual odometry faces significant challenges that must be overcome. One of these challenges is handling variable lighting of the changing lighting conditions, which affects the quality of the captured images and, thus, the precision of the movement estimation. Noise or occlusions can also present additional difficulties in tracking or matching between reference points in the images.

To face these challenges, researchers have been exploring advanced approaches, such as using CNNs to extract the relevant features from the images and filtering algorithms to reduce the impact of the noise [27]. Furthermore, sensor fusion between different sensors, such as cameras and IMUs, has been used to improve the precision of the visual odometry, compensating for the individual limitations of each sensor [28].

Another crucial aspect of visual odometry is the precise calibration of the intrinsic and extrinsic parameters of the camera. An unsuitable calibration may lead to significant errors in the movement's estimation, compromising the system's reliability. Therefore, it is essential to perform precise and rigorous camera calibration processes.

Concluding, visual odometry is a powerful technique to estimate the robot's movement based on visual information. Although it presents notable advantages, such as obtaining detailed information about the environment, visual odometry also faces challenges related to variable lighting, noise, and occlusions. However, with the continuous advance in research in computer vision and image processing algorithms, visual odometry is still used, and it is a promising area for the development of more precise and more robust robotic navigation systems.

2.1.4 Genetic Algorithm

The genetic algorithm is an optimization technique inspired by biological evolution and is widely applied to several problems. It is based on natural selection, where candidate solutions are treated as individuals among a population, and genetic operators such as selection, breeding, and mutation try to find increasingly better solutions along generations [29].

In the context of obstacle avoidance, genetic algorithms have been explored to find safe and efficient robot trajectories in unknown and dynamic environments [30]. The genetic algorithm approach allows for an exhaustive search space exploration, making discovering unintuitive and robust solutions possible.

A fundamental aspect of applying genetic algorithms in obstacle avoidance is the representation of the candidate solutions. One common way is the codification of trajectories as a sequence of coordinates or angles, allowing the generation and manipulation of different trajectories through genetic operators. Furthermore, predictive genetic codification techniques have been developed to optimize a sequence of the robot's actions directly, considering the environment and the existing obstacles [31].

One of the main challenges of applying genetic algorithms to obstacle avoidance is the definition of a suitable fitness function that quantifies how good the individual is at obstacle avoidance. Metrics such as traveled distance, time of execution, safety, and trajectory smoothness and commonly used to evaluate the performance of the candidate solutions [32].

To improve the efficiency and the convergence of the genetic algorithm, strategies such as elitism, where the best individuals are unmodified in the subsequent generations and the diversity is controlled, where mechanisms are used to preserve the genetic diversity of the population, have been applied. Furthermore, the incorporation of specialized knowledge, such as obstacle avoidance heuristics based on rules, can improve the performance of the genetic algorithm by guiding the search for the best solution through a more promising path [33].

Although genetic algorithms present several advantages in obstacle avoidance, it is essential to emphasize that they also have limitations. The population size and the number of generations required to reach an acceptable solution can be high, which results in considerable computation time and power. Furthermore, the chosen representation can limit the search space exploration, and there is no guarantee that the algorithms find the globally optimal solution due to the stochastic nature of the evolutive process [34]. However, with suitable strategies for the algorithm's parameter configuration and a good balance between exploration and exploitation, it is possible to obtain promising results in obstacle avoidance through genetic algorithms.

2.1.5 Fuzzy Logic

Fuzzy logic is a mathematical system based on the theory of fuzzy sets, which handles inherent uncertainty and lack of precision from real-world problems. Contrary to classic binary logic, where a proposition is true or false, fuzzy logic allows the representation of continuous truth values in an [0,1] interval. This characteristic allows for more suitable modeling of complex phenomena which absolute propositions cannot easily describe.

In fuzzy logic, linguistic variables describe vague and unprecise terms and concepts in real-world systems. These variables can assume different values within their universe of speech, representing a given property's degree of relevance or intensity concerning a specific fuzzy set. These fuzzy sets are defined by relevance functions describing the relationship between their linguistic variable value and degree of relevance [35].

Fuzzy logic offers a flexible and adaptable approach to model and control complex systems, such as ASVs. In obstacle avoidance, fuzzy logic can be applied to make real-time decisions based on sensor data, allowing vehicles to avoid collisions and navigate safely in challenging environments [36].

A practical example of fuzzy logic application in autonomous is the thrusters' velocity control. It is possible to adjust thrusters' velocity according to the environment and the presence of obstacles using a set of pre-defined fuzzy rules. These rules must derive from specialized knowledge or even be learned from machine learning algorithms [37].

Furthermore, fuzzy logic can be combined with other control methods, such as PID (Proportional-Integral-Derivative) controllers or optimization algorithms, to improve the performance and efficiency of obstacle avoidance systems even more. The advantage of the fuzzy logic approach is its capability to handle the uncertainty and lack of precision inherent to sensors and the environment, allowing the vehicle to make robust and adaptable decisions even in harsh conditions [38].

However, it is essential to highlight that the conception and implementation of fuzzy logic systems require specialized knowledge and experience due to the complexity of the design process and the definition of the fuzzy sets, and the syntony of the parameters and their control rules. Furthermore, fuzzy logic's effectiveness depends on the quality of the sensor data and the suitable selection of linguistic variables and fuzzy sets.

2.1.6 Adaptive Velocity Obstacle Avoidance (AVOA)

The adaptative velocity obstacle avoidance method introduces the innovative concept of protective zones to avoid collisions in ASVs [6]. This method was specifically designed to be implemented in ASVs. However, it can be applied to other areas. This approach considers the direction and magnitude of the ASV's velocity vector, allowing it to predict and avoid imminent collisions in the near future. This method is especially relevant by considering the obstacles' velocities, contributing to safer and more efficient navigation.

Protection zones are defined from the projection of the line of sight to the obstacle in the 2D space while introducing a safety radius around the obstacle. This method ensures a margin of safety suitable for obstacle avoidance. The protection zone is then moved and translated in the function of the obstacle's linear velocity, guaranteeing the path planning to consider the obstacles' movement in real-time.

An important aspect is identifying the non-imminent collision zone, which is calculated by removing the area in which the distance to the protection zone is less than the minimum distance allowed from the obstacle while considering the safety radius. This non-imminent collision zone allows the vehicle to avoid obstacles safely, granting a reasonable safety margin even in high-speed situations of evasive maneuvers.

For better comprehension, figure 2.5 visually shows the concept of protection zones. The lines of sight from the objects in the 2D space can be observed, as well as their respective protection zones represented by bounded areas. These areas delimit the zones where the ASV can move without the risk of an imminent collision, allowing for a wide range of options for safe trajectories.

It is important to note that the protection areas define the limits for the vehicle's movement, thus granting safe and collision-free navigation [6]. However, it is vital to consider the specific

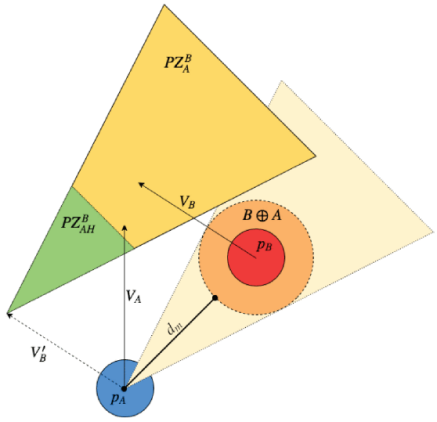


Figure 2.5: Representation of the protective zone concept [6]

kinematic restrictions of the ASV. Not all safe velocity vectors can be followed due to the limitations imposed by the ASV's kinematic model. However, AVOA includes a filter to remove impossible velocities, granting all achievable velocities are considered.

Selecting the optimal velocity vector is crucial to ensure efficient and safe navigation. The AVOA method uses optimization techniques to find the best solution, minimizing the deviation between the pre-defined speed while maintaining a safe and obstacles free path. Among the optimization techniques applicable to the AVOA, particle swarm optimization [39], simulated annealing [18], and genetic algorithms [29] can be highlighted. These approaches allow the finding of effective and adaptative solutions for different scenarios and navigation conditions.

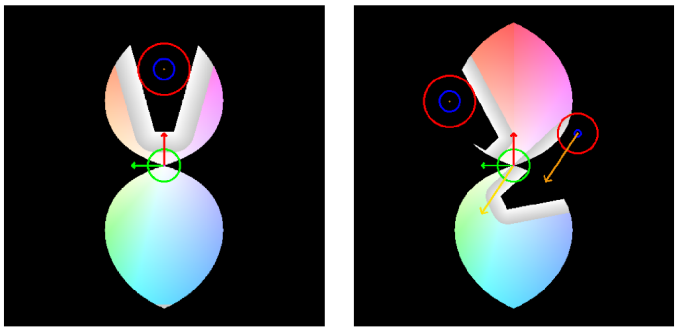


Figure 2.6: Possible velocity vectors constrained by vessel's kinematics (left) of a single static obstacle and (right) with a static and a dynamic obstacle [6]

Figure 2.6 shows the possible velocity vectors, considering the kinematic restrictions of the ASV. On the left side is the case of a single static obstacle, while on the right side is a static and a moving obstacle. Analyzing these velocity vectors makes it possible to identify safe and viable trajectories for ASV navigation around the obstacles.

In Daniel Campos' work [6], a comparison between the AVOA method and the Time Elastic Band technique (TEB) [40] was made. Results show that AVOA had a 58% improvement in travel time compared to TEB, besides generating smoother trajectories. In some scenarios, TEB did

not achieve the final destination of the trajectory, while AVOA was always successful. However, it is essential to note that AVOA is susceptible to local minima, which can lead to sub-optimal solutions. Furthermore, AVOA lacks consideration of angular velocities when predicting and projecting the protective zones. Angular velocities play a crucial role in the curve maneuvers of vessels, and it is essential to predict their movement to prevent a collision.

Despite the limitations, the AVOA method represents a significant advancement in ASV autonomous navigation, providing an adaptative and efficient approach to obstacle avoidance. With future research, it can be expected that AVOA could be applied to various scenarios and contribute to the safety and efficiency of ASV autonomous navigation.

2.2 Scenario Representation

Scenario representation in robotics refers to how a robot's software represents and processes information about its current state and the tasks it needs to perform. This may include information about the robot's surroundings, goals and objectives, and any constraints or requirements the robot must consider as it carries out its tasks. Scenario representation is an essential aspect of robot planning and decision-making, as it allows the robot to consider a range of possible courses of action and choose the most appropriate one based on its current situation.

2.2.1 Grid-based Methods

Occupancy grids are widely used ways to represent the environment in robotics. This approach allows for the mapping of the environment in discrete cells, where each cell is classified as occupied by an obstacle or free. In this space representation, the grid can vary between a 2D, 2.5D, or 3D representation, depending on the environment complexity and system requirements [41]. An example of a 2D representation is shown in figure 2.7.

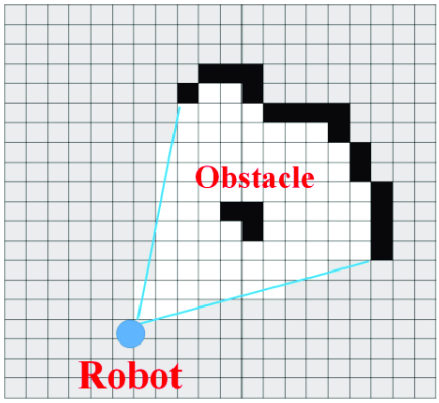


Figure 2.7: Example of a robot and an obstacle in a 2D occupancy grid [7]

This grid-based representation offers several advantages for navigation and path planning. One is the ability to feed obstacle avoidance algorithms with a real-time updated map. As the vehicle collects data from the scenario, the occupancy map can be updated and corrected continuously, allowing a dynamic perception of the environment.

The map's resolution is a crucial factor to be considered because it directly influences the precision and effectiveness of the representation. A high-resolution results in a more detailed map but increases the required computational power as it grows quadratically. On the contrary, a low resolution leads to the loss of information. Therefore, a good balance between grid resolution and computational power consumption must be achieved to meet the specific application requirements.

In the context of ASVs, the 2D representation of occupancy grids is vastly adopted due to the characteristics of the aquatic environment [42]. In these conditions, most obstacles are located on

the same 2D plane, making a 2D representation enough to capture relevant data and allowing for safe and efficient obstacle avoidance.

One of the main challenges of the occupancy grids is the computational power demand, especially in big maps. The processing of an individual cell can take a significant execution time, which can hurt the real-time performance of the system. To overcome this problem, solutions such as Sang-Il Oh's, where superpixel-based cell clustering is used to reduce the required computational power without compromising the precision of the map [43], improving the computational speed by around 38.9% and speed by approximately 12% over Mekahanachas method [44].

Despite occupancy grids' simplicity and efficiency, they have some limitations. Because they use a discrete representation, they cannot provide a smooth representation of curved or irregular surfaces. Furthermore, grids can be affected by sensor noise, resulting in false positives or false negatives in obstacle detection. Therefore, additional filtering and processing techniques are required to handle these challenges to improve the precision and reliability of the occupancy grids.

2.2.2 Polygonal Maps

Polygonal maps are a simplistic and efficient way of representing the environment in the 2D plane. They are widely used in robotics to map and navigate through complex scenarios. Contrary to grid maps, polygon maps allow for a more flexible and detailed representation of the environment since the positioning of the polygons is not constrained to a grid.

The creation of the polygons occurs through the projection of the boundings of the obstacle on the XY plane. This approach generates convex polygons that perfectly adjust to the contours of the objects in the environment. Figure 2.8 demonstrates the polygon creation process based on the obstacle borders, despite the sensor noise.

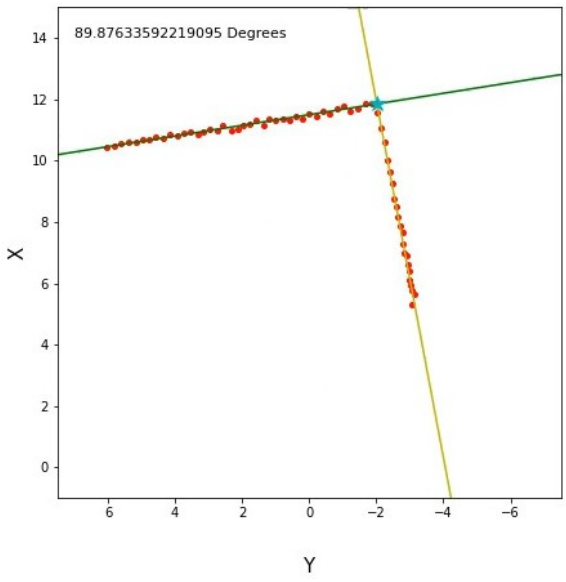


Figure 2.8: Approximation of a noisy 2D LiDAR measurement into two lines

One of the methods used to calculate the polygon lines is the technique proposed by L.J Letecky in their work [45]. This approach involves finding the first-order function that better adjusts to the data points corresponding to the obstacles borders. This process ensures the precision of the polygon representation, making them a reliable choice for navigation and obstacle avoidance.

Furthermore, another technique is the creation of bounding boxes [46]. These bounding boxes are rectangles that overestimate the obstacle's size, simplifying its representation. The advantage of this approach is the reduction in computational complexity, making the processing of the polygons more efficient in terms of execution time.

Polygon maps are especially useful in scenarios that require detailed maneuvers, such as the autonomous vehicle's docking of obstacle avoidance in complex environments. They provide a precise and detailed representation of the environment, allowing obstacle avoidance algorithms to make decisions based on reliable data.

However, polygon maps have some limitations. Because they are based on a fixed representation of the environment, they do not handle dynamic changes in the shape of objects or moving obstacles. Furthermore, due to the high level of detail, polygons require a considerable amount of memory to store the data and computational power to process data in real time.

2.2.3 Voronoi Diagrams

Voronoi diagrams are potent tools in robotics path planning and obstacle avoidance. They represent an efficient technique to map an environment while identifying the safe and efficient paths for the vehicle to follow in a complex scenario. An example of a Voronoi diagram is shown in figure 2.9.

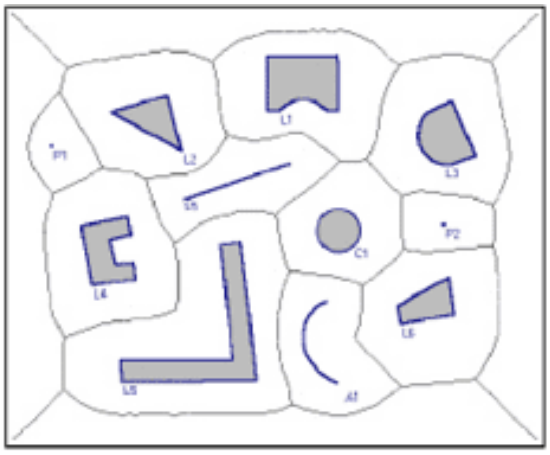


Figure 2.9: Voronoi diagram example [8]

A Voronoi diagram can be assembled using the fortune algorithm, which sweeps a line across the 2D plane and analyzes input data, updating the diagram each time instance. This algorithm uses a binary tree data structure, allowing fast and efficient searches among the diagram [**?**].

The main feature of Voronoi diagrams is that they divide the scene into regions, where each region is defined by a set of the closest points to a determined object. The resulting lines from the Voronoi diagram are equidistant to the objects that originate them, making them a suitable representation of the boundaries between objects' influence regions. These lines can be considered safe paths for navigation once they are as distant as possible from the obstacles.

One of the advantages of the Voronoi diagrams is their information representation efficiency. They are compact because they only store the seeds that define the regions' origin points in memory. This makes them a suitable choice to represent spatial information with memory restrictions. Furthermore, Voronoi diagrams are highly adaptable to dynamic environments once they can easily be updated as new data is collected.

However, building Voronoi diagrams can be computationally expensive, especially in scenarios with several objects and input points. The fortune algorithm requires significant time and resources to calculate the correct region boundaries. Furthermore, the precision of the Voronoi diagram depends on the quality of the input data, including the precision of the objects' measurements and the spatial resolution of the input data. Noise or unprecise data can strongly affect the quality of the resulting diagram and thus influence the navigation paths of the vehicle.

2.2.4 Point Clouds and Voxels

Point clouds are a combination of non-structured points in the cartesian 3D space. These points are usually the result of LiDAR sensors or depth cameras and are widely used in the navigation and reconstruction of tridimensional environments.

Due to the non-structured nature of point clouds, there is no data about vertice connectivity. Therefore, there is no explicit impression of the subjacent surface on which points were sampled, and algorithms similar to image processing cannot be directly used. However, a local structure can be explored to infer surfaces properties and create an approximated representation of the objects.

A common approach to handling point clouds is interpreting the data points using voxels [47]. Voxels are tridimensional cells that divide the space into a regular grid and represent the presence or absence of points inside each cell. This grid representation allows for data reduction and simplification, making voxels more suitable for real-time processing and analysis. Figure 2.10 shows an example of the voxel representation of a point cloud.

The use of voxels in point cloud representation offers some advantages [48]. Firstly, it allows for a reduction of the data size, making them more efficient in terms of storage and processing. Furthermore, the voxel representation is robust relative to noise and point density variation once each voxel is analyzed independently. This approach also simplifies the detection and segmentation of objects in the point cloud once the data is structured in discrete cells.

Another way of representing point clouds is through octrees. Octrees are tree structures where each node has eight children, hierarchically dividing the space. This representation is especially useful when an adaptative representation with different detail levels in specific point cloud regions is desired. The construction of the octree involves subdividing the voxels into eight smaller equal-sized voxels until a stop criterion is met, such as maximum subdivisions or the minimum voxel

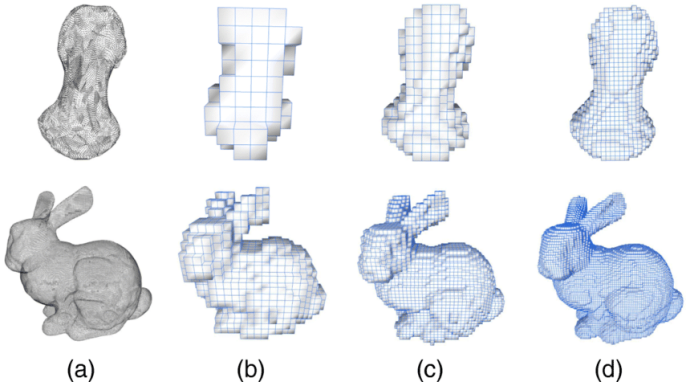


Figure 2.10: Voxel representation in of the two original objects (a) in different resolutions [9]

size. This way, octree representation allows for an efficient allocation of computational resources, concentrating them in regions of interest.

Despite their advantages, voxel and octree representations have some disadvantages. The most relevant is the loss of detailed information about the objects' shape and surface since both representations are based on discrete cells. Furthermore, the size of the voxel or the resolution of the octree can directly influence the quality and reliability of the representation. Therefore, a suitable choice of resolution parameters is crucial to balance computational efficiency and data detail.

2.2.5 Topological Maps

Topological maps are a type of widely used type of map in robotics to represent the topology of an environment in an abstract way [49]. In robotics, topology refers to how different locations within a space are connected. It describes the relationships between different parts of the environment rather than their exact positions or shapes. While conventional maps provide precise information about spatial coordinates and the object's physical shape, topological maps represent the connectivity relationships between different parts of the scenario [50].

One of the central elements of topological maps is the nods, which represent significant regions or locations in space. These nodes can be interpreted as reference points, areas of interest, or specific locations with distinct features. The edges establish the connections between two nodes and represent possible trajectories or paths.

It is common to represent obstacles in topological maps by using polygons or bounding regions that limit the areas occupied by those objects [49]. These polygons can vary in shape and size, depending on the features of the obstacle. The left side of 2.11 shows an example of a topological map with its obstacles represented by polygons. Another way of representing a topological map is by representing obstacles by points rather than polygons. In this case, edges connect to the landmarks nodes on every observation [51], as can be visualized on the right side of figure 2.11

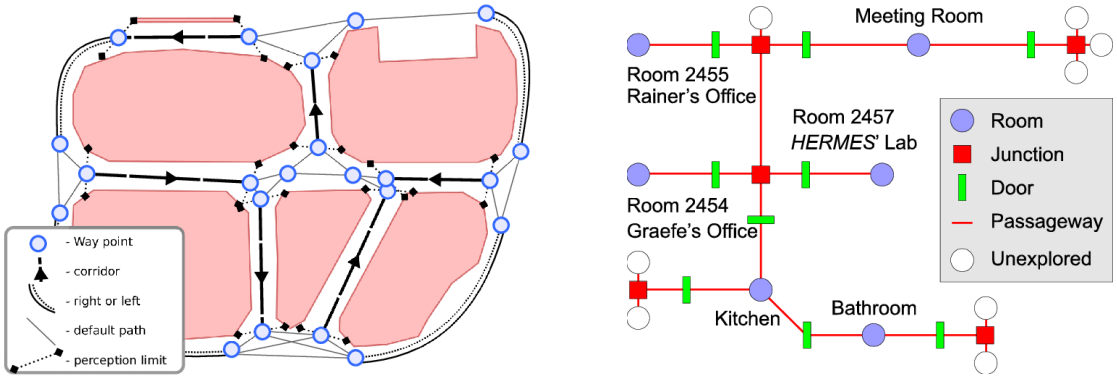


Figure 2.11: Two topological maps with different obstacles representation

Different techniques can achieve topological map building. A common approach is using simultaneous mapping and location (SLAM) algorithms [50], which allow the robot to build the map as it explores and interacts with the environment. Furthermore, the fusion of data from different sensors, such as LiDAR, cameras, or sonars, can improve the quality and precision of topological maps.

The main advantage of topological maps is their capability to abstract from spacial information, robustness to changes, scalability, and human interpretability. By only representing essential information about connectivity, these maps can describe various environments and favor autonomous navigation. It is especially useful in scenarios with metric precision is not crucial or cannot be easily obtained and efficient obstacle avoidance is required. Topological maps' flexibility and ease of updating make them suitable for dynamic scenarios where the environment is constantly changing. Furthermore, the abstract nature of topological maps allows for a compact representation, resulting in lower memory and computational power demand, making them suitable for real-time applications.

Despite topological maps' advantages, they have some disadvantages. The main one is the lack of precise metric information, which can be problematic in scenarios requiring high precision, such as tasks involving interaction with the environment. Furthermore, the abstract representation can limit the capability to make detailed decisions compared to metric or probabilistic maps.

2.3 Critical Review

This section resumes the conclusions from the literature review of the state-of-the-art methods described in this chapter.

The most promising approach to obstacle avoidance and path planning is AVOA. The reason why is that this algorithm was specially developed to be deployed in ASVs. This method is also suited for static environments and dynamic scenarios where obstacles move. Because of the concept of the definition of protective zones, this algorithm is the safest of all mentioned in the previous section. To tackle the detected AVOA's downside, a more predicted behavior can be implemented, although the result presented in the reviewed papers already had good results. For the inputs of this algorithm, a dock detector [52] and a vessel detector and tracker [53] should be used since they are already prepared for this type of application.

For the representation of the scenario, the use of topological maps could benefit the applications ASVs are usually submitted. These maps perform exceptionally well when compared to other methods. In the maritime environment, adverse weather conditions are present, for example, rain, fog, waves, or even the Sun's reflection. In this case, sensors do not perform optimally, thus providing less reliable information. Topological maps are more forgiving in this regard and thus can provide better results in navigation. The attribution of semantics to the environment and objects, in general, allows the system to understand the significance of the various elements it interacts with and make informed decisions about avoiding obstacles and planning a path.

Chapter 3

A Semantic Map Representation for Obstacle Avoidance

3.1 Introduction

A good representation of the surrounding scene is vital to a robust navigation system. Obstacle avoidance algorithms require some form of a map containing the obstacles, boundaries, and references to perform their task correctly. Motivated by the fact that the maritime industry is tending to autonomous solutions, this dissertation aims to develop a semantic mapping solution based on topological maps to represent autonomous vessels' surroundings. This representation can then be used for obstacle avoidance and inspection tasks.

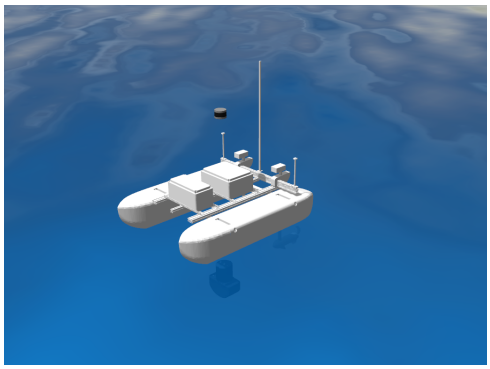
This work used a Gazebo 3D simulation of the SENSE ASV and the Nautilus ASV to develop and test the proposed algorithm. These two different test vehicles were used to test and show that the method is agnostic to the deployed platform. The Nautilus ASV, illustrated in figure 3.1b, is equipped with the necessary sensors, including:

- Camera - *HIKROBOT MV-CA016-10GC - Frame Rate: 78.2Hz, Resolution: 1440 × 1080, Equipped with 12.5mm focal length lens* horizontal;
- LIDAR - *Ouster OS1-128 - Frame Rate: 10Hz, Range: 200m, Range Accuracy: 0.05m, Field of View: 45º, 360° horizontal and 45° vertical;*
- IMU - *XSens MTI-630 - Frame Rate: 520Hz, Angular Accuracy: 0.2°/0.5°;*
- GPS - *Holybro F9P Helical - Frame Rate: 20Hz, L1/L2 RTK, Accuracy: 0.01m horizontal, 0.015m vertical.*

Additionally, the SENSE ASV simulation, illustrated in figure 3.1a, uses a similar sensor setup, described by:

- Camera - *Mako G-125 - Frame Rate: 30Hz, Resolution: 1292 × 964, Field of View: 80°* horizontal;

- LIDAR - *VLP-16 - Frame Rate: 4Hz, Range: 100m, Range Accuracy: 0.03m, Field of View: 360° horizontal and 30° vertical;*
- IMU - *MTi-30Xsens - Frame Rate: 200Hz, Angular Accuracy: 0.2°/0.5°;*
- GPS - *Swift Navigation - Frame Rate: 20Hz, L1/L2 RTK, Accuracy: 0.01m horizontal, 0.015m vertical.*



(a) Gazebo 3D simulation of the SENSE ASV

(b) Nautilus ASV

Given that the test environment is for offshore inspection, the set of objects composed of vessels and docks was chosen because they are relevant elements in this specific scenario. The proposed method was implemented with the intent of being used to feed the AVOA algorithm with the scenario information to avoid obstacles. This obstacle avoidance technique requires a representation of each object's pose and velocity. Therefore every detector should provide at least these two features. The vessel detector takes an image and identifies and classifies different types of vessels. Besides the image collected by the monocular camera, a point cloud retrieved from LiDAR is used to estimate the XYZ position in the environment. Contrary to this, the dock detector uses only LiDAR data to detect and classify docks and their respective pose.

With the odometry information and the previously described scenario landmarks, a pose graph can be built to describe the surroundings. Odometry nodes are connected sequentially, and the landmark nodes are connected to the last odometry node upon observation. Once the ASV loses sight of that landmark and re-observes it, a new connection is made, and its pose is corrected considering all the involved covariances, providing both a scenario representation and a localization method. This data is required for good functioning of the obstacle avoidance algorithm.

Considering the core functionalities mentioned above to perform autonomous obstacle avoidance with a topological map, a three-stage architecture was adopted as described in figure 3.2.With this methodology, the proposed solution is scalable and adaptable: more detectors can be added and used in other types of vehicles (e.g., aerial or terrestrial).

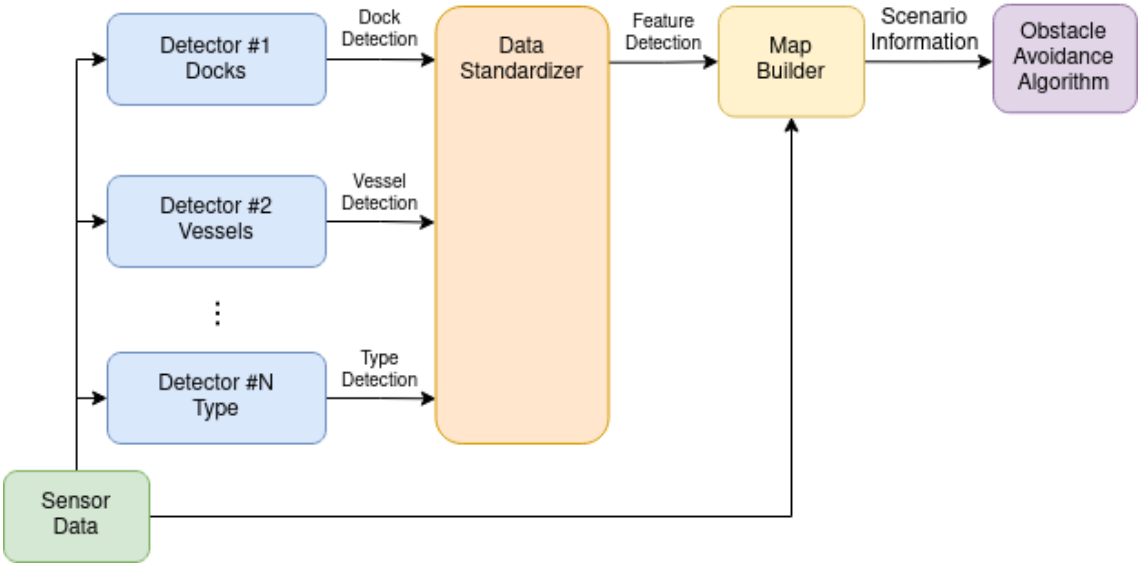


Figure 3.2: Proposed architecture scheme for semantic mapping and obstacle avoidance using multiple detectors.

3.2 Detectors

3.2.1 Vessel and Buoy Detection

Vessels are the most common elements in the maritime scenario, both in open-sea or in ports. Collisions being one of the leading causes of maritime accidents, it is vital to detect them reliably. However, these vehicles can be moving in and out of frame, leaving them undetectable for some time (occlusions). For this reason, it is mandatory to predict their position over time. Buoys are common in ports, offshore wind farms, coastlines, and other places, making them relevant landmarks to add to a map.

Initially, the detector used YOLOv5 to detect and track the referred objects. This computer vision algorithm allows the classification of objects in an image with extreme speed and efficiency, making it suitable for real-time application. However, the detector was upgraded to use YOLOv7, which promises to be 120% faster for the same average precision [10], as seen in image 3.3.

The Datasense@CRAS dataset was used to train the new YOLO model, following a similar procedure as Duarte [53]. However, it was discovered that the class occurrences were unbalanced, as shown in figure 3.4. Therefore, data augmentation was used when training the new YOLO neural network to compensate for the dataset's imbalance and thus further improve the model's accuracy. Data augmentation was achieved by rotating, re-scaling, and mirroring elements from the least represented classes. Many elements of the *Bulk carrier* class present in the dataset were extracted from frames of the same video. Therefore, some of them were removed, increasing the time between each consecutive frame and restoring some of the class balance of the dataset.

When implemented in the ROS framework, the detector outputs an array where each element is the detected class of vessel or buoys, its confidence, its pose with the respective covariance, and its bounding box. Figure 3.5 shows an example of the successful detection of multiple vessels.

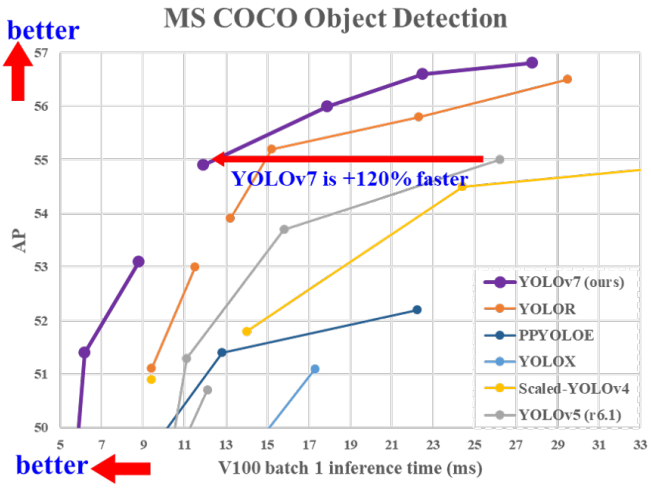


Figure 3.3: Average precision comparison between different YOLO versions [10].

Since the detector uses only visual information to recognize vessels and buoys, an additional stage is required to estimate the XYZ position of the object. Data fusion between LiDAR data and the original image is used in this context. The LiDAR-generated point cloud is projected over the camera plane, resulting in a depth map aligned with the original image. By considering only the points inside each detected object's bounding box, it is possible to use a K-means algorithm [54] to filter the background noise and calculate the average depth of the vessel or the buoy.

This data fusion approach provides a better tridimensional estimate of the detected objects, as the visual information is complemented with the depth information provided by LiDAR. This way, the proposed method overcomes the limitations of the monocular camera, which include difficulties in estimating the actual position of objects. Furthermore, using a K-means algorithm allows efficient background noise filtering, granting more reliable results in estimating the average depth of vessels and buoys.

It is important to note that this detector's spacial information is relative to the camera frame. In a later section, it will be explained how it was possible to retrieve the third dimension of the measurements so they can be helpful for a local representation.

3.2.2 Dock Detection

Contrary to vessels, docks are only present in ports and are permanently stationary. Due to the lack of space to maneuver in harbors, docks become a valuable landmark for a map. The precision and accuracy of the dock's position are critical to prevent collisions, especially when docking. This detector is also capable of analyzing the occupancy of the pier. This is relevant because there is a high probability of a vessel docking there. If that is the case, it simplifies the inference process on whether the vessel is moving.

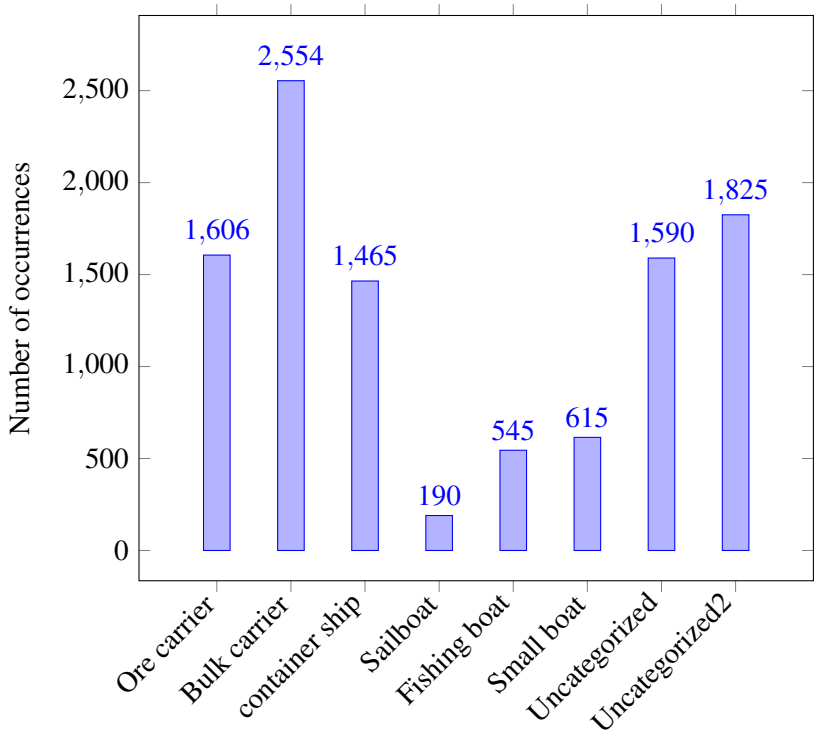


Figure 3.4: Distribution of the occurrence of each class in the entire dataset.

The dock detector was implemented by following Pereira's methodology [52]. This detector relies on LIDAR data to perform its task. In the pre-processing, it filters the retrieved point cloud to remove the horizontal plane and to compensate for eventual undulation using the IMU's data. Then the detector segments the point cloud into voxels and uses a neural network to evaluate whether one or more docks are present or more in the frame. If there are docks in the frame, the model classifies them according to their shape. Figure 3.6 shows an example of a dock detection visual output.

This implementation outputs an array with the information on the detected objects. This information includes the dock type and its centroid. The detector also puts a bounding box around the corresponding data points of each dock on the RViz data visualizer with its confidence.

3.2.3 Detector Standardization

One of the core elements in any topological map is the features of the environment. Features are relevant elements of the scenario used as landmarks in the map and for relative navigation. These physical or structurally relevant scenario elements are used as landmarks on the map and have a crucial role in autonomous navigation. These features' detection and precise feature extraction are essential for assembling a reliable and precise map.

Specific-purpose detectors, such as those discussed in the last section, can be used to identify relevant environmental elements to obtain these features. However, a challenge arrives: each detector provides its heterogeneous data with a different data structure, which can complicate

Figure 3.5: Example of successful detection of a boat.

the consistent and standardized interpretation of this data by the map builder. Therefore, the standardization procedure is required to ensure the features are interpreted uniformly regarding the originating detector. This standardization makes the map builder system adaptable and scalable, incorporating different detectors and their specific output data structures.

The adopted standard data structure to represent the detected features has several vital elements. Firstly, the feature's pose provides information about position, orientation, and velocity. This data is fundamental for precise mapping and localization in the context of a robotic system. Furthermore, a timestamp is crucial since detectors can operate in different frequencies. Using suitable timestamps allows the correct synchronization between data regarding different detection time instances. Ensuring precise time alignment is vital because it not only grants proper data coherence but also facilitates the temporal interpolation of information and allows the calculation of further data. The velocity is calculated using these timestamps along with the last recorded pose of each obstacle since the high frequency of operation provides a good approximation of the position derivative in the equation:

$$V = \frac{D}{\Delta t}, \tag{3.1}$$

where V is the velocity vector, D is the displacement vector, and Δt is the time interval between the two measured poses used to calculate the displacement or, in this case, the difference between the two consecutive timestamps.

Another relevant aspect is the reference frame, which specifies the original coordinate system on where the data is expressed and their spatial relations. Sensors are located in different physical positions on the vehicle or use different coordinate systems, and that information is crucial to transform the data into the map reference frame suitably. A reference frame transformation is mandatory to ensure the features are correctly represented and aligned with the map. This transformation consists of performing a translation and a rotation on coordinate system rotation. Figure 3.7 shows an example of a transform between the frames.

Feature covariance is another piece of information present in the data structure. It reflects

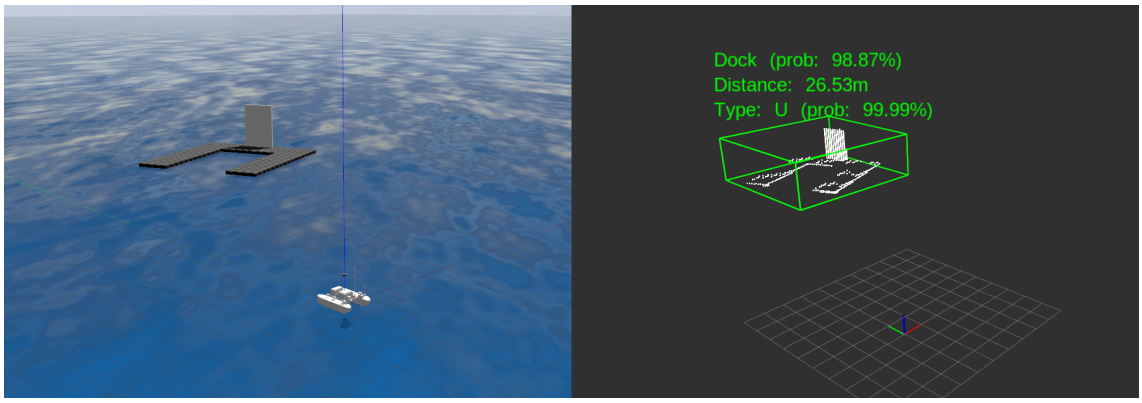


Figure 3.6: SENSE ASV and a dock in the Gazebo simulator (left) and successful dock detection visual data representation (right).

the uncertainty associated with the detection of the feature, and it is a statistic that expresses the variability of the data. In the context of the topological map, the covariance provides a feature's quality information.

Finally, the feature type classifies each element in its specific type or subclass depending on its properties or intrinsic characteristics. This data helps perform analysis and specific operations related to each feature type.

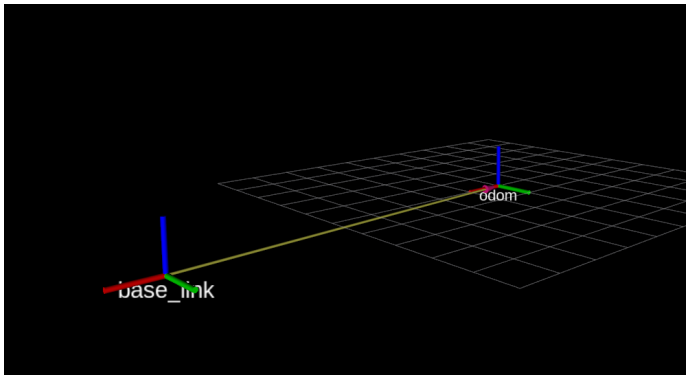


Figure 3.7: Example of a reference frame transformation from the robot frame into the odometry frame.

By standardizing the feature data with the presented structure, the map builder can suitably process, analyze and represent the scenario features in a tridimensional virtual environment while making the system adaptable and scalable.

It is important to note that the sensors must be calibrated to achieve the most reliable and accurate results. With a correct calibration, the pose and velocity estimates would stay consistent with their actual values, and the classification algorithms would improve accuracy. The calibration of the sensors was performed similarly to the described procedure in Campos' work [2]. The cameras were calibrated using a chessboard to calculate the extrinsic parameters with the corner extraction method [55]. The intrinsic parameters of the LiDAR were pre-calibrated. The single value decomposition of a set of 3D correspondences was used to calibrate the relation between

the LiDAR and the camera reference frames [56]. The GPSs and IMU frames were calibrated by manually measuring them in relation to a reference frame.

3.3 Scenario Representation

The chosen approach to implement a topological map was a pose graph. In this representation, each node in the graph is related to odometry or a feature. Both nodes and edges can only be added if they meet the respective criteria, which are different for each node type. The rules can be adjusted to increase the map's resolution or reduce the computational power requirements.

The pose graph was implemented following a two-part structure: the pose graph builder, which is subdivided into odometry and features, and the visualization. This allows for better modularity in the system and the possibility of using the map data for different applications than obstacle avoidance. Figure 3.8 illustrates this architecture.

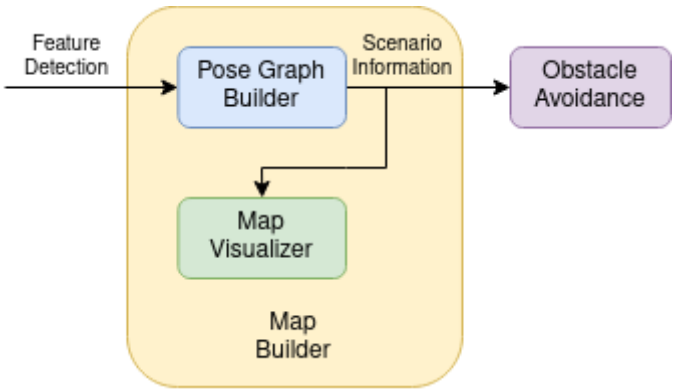


Figure 3.8: Architecture used to implement the topological map.

3.3.1 Odometry

Odometry is a crucial component in autonomous robotics localization, mapping, and obstacle avoidance. It describes the cumulative estimate of the vehicle's pose over time. In the context of this research work, the odometry system is provided to supply that critical information.

As discussed in the previous section, odometry is calculated relative to the vehicle's internal reference frame, and thus, it must be transformed into the map reference frame. This transform is necessary because the odometry's reference frame begins when the robot is powered up, which means the initial position in the map can vary between missions within the same map.

This odometry system used in this work relies on the Extended Kalman Filter (EKF) with the IMU (roll, pitch, yaw, and angular velocity values) and GPS data as inputs. The integration of the IMU data in relation to its initial state is used for the prediction state of the filter. The GPS data is converted to Universal Transverse Mercator (UTM) using the transformation defined in the United States Geological Survey (USGS) Bulletin 1532 and used in the correction stage of the EKF. By

projecting the odometry into the UTM projection, it is possible to obtain a reasonable estimate of the ASV's position in the world and its evolution from the first GPS measurement [2].

It is important to note that, due to the cumulative nature of odometry, the covariance associated with the pose estimate increases with each new calculation. However, incorporating periodical GPS readings makes it possible to decrease the covariance, making odometry more robust. By comparing the GPS measurements with the odometry's pose estimate, it is possible to perform a correction, reducing the covariance. Once at least three features are detected, it is possible to use triangulation techniques to estimate the vehicle's position relative to those features with increased accuracy. This allows for an improvement in the quality of the mapping and localization of the robot in the scenario.

In the specific context of this work, odometry nodes are added to the map according to some criteria. One of these criteria is the minimum time elapsed since adding the previous node. This prevents the excessive addition of nodes and reduces the overall computational complexity of the system. The minimum traveled distance by the vehicle is also considered when adding new nodes. This restriction implies that only significant movements are captured, avoiding map pollution with redundant information.

Finally, ensuring a uniform distribution of odometry nodes, thus avoiding low-density areas, is achieved by establishing the maximum distance traveled by the vehicle and adding new nodes if the vehicle's speed exceeds a certain threshold, regardless of whether the last two criteria are met.

These rules and criteria for adding odometry nodes in the map provide precise control over the map's resolution and, thus, the computational demands associated with the processing and updating of the map over time. This allows for efficient optimization of the mapping system, making it more suitable for autonomous robotics.

3.3.2 Features and Landmarks

In topological maps, distinct environment features are used as landmarks to improve the localization and obstacle avoidance systems. These features can be static or dynamic and play a crucial role in mapping and autonomous navigation.

As discussed in the last section, provided feature data, such as pose, are represented in the correct coordinate frame. However, parameters specific to each detector must be configured to handle different uncertainties and behaviors of their features. These parameters are crucial in the features' data fusion and covariance estimate.

One of the critical parameters is the minimum feature distance. Detectors usually have uncertainties about the feature's position, which results in variation at each time instant. If this deviation makes the feature's position land inside the covariance ellipse, its position should be recalculated with the average between the last two estimates. However, if the deviation is lower than the minimum feature distance, the feature's position is assumed to be the same and, thus, not recalculated, saving computational power. This process can be described by the flowchart in figure 3.9. The feature's position can also be predicted based on its last velocity estimate. On the other hand, if

any other feature's covariance ellipsis does not contain the detected feature's position, it is considered a new feature and added to the map. Another parameter is the feature ID. Some detectors can detect moving features (e.g., a boat). These features' unique ID allows for their tracking over time[57], which is crucial for obstacle avoidance and path planning.

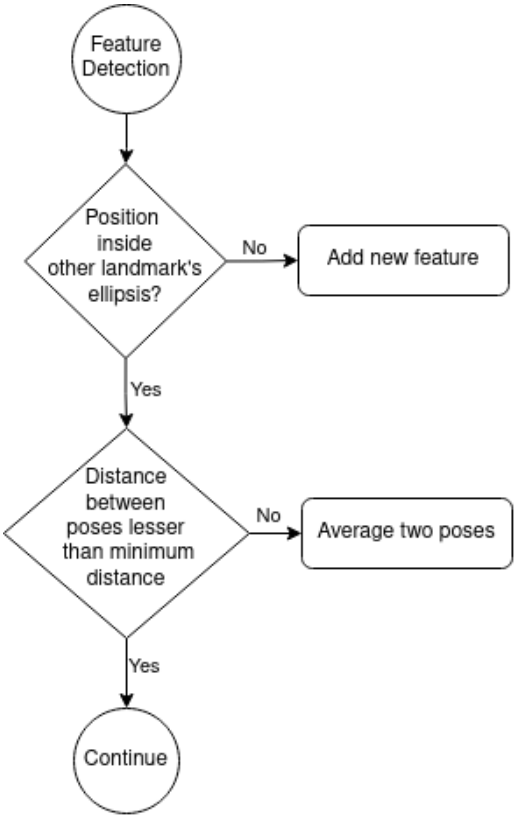


Figure 3.9: Behavior of the algorithm when observing a new landmark.

Features and odometry covariances fusion is a critical aspect of obtaining a precise representation of the environment over time. Features' covariances are calculated with a weighted sum of the two mentioned covariances, leading to a better covariance estimation. It is important to note that this process is only possible due to the assumption of the position variables following a Gaussian distribution. By definition, the Kalman filter used in the provided odometry system already assumes that the sensor data follows a Gaussian distribution, and the fact that these sensors are used to calculate features' positions makes it valid to assume them Gaussian variables.

Covariances' correct orientation is essential for a suitable integration of features' data. Odometry covariances are expressed in the vehicle's reference frame and must be transformed into the map reference frame before combining it with the feature's covariance. For this reason, it is necessary to rotate the odometry matrix covariance using a suitable rotation matrix. Covariance rotation is critical in granting that features' data are correctly aligned with the global scenario. Equation 3.2 proves that a rotation matrix can be applied to achieve this goal.

$$
\begin{aligned}
\mathbf{C} &= \mathbf{E}(\mathbf{X}\mathbf{X}^T) - \mathbf{E}(\mathbf{X})\mathbf{E}(\mathbf{X}^T) \\
\mathbf{C}' &= \mathbf{E}(\mathbf{R}\mathbf{X}\mathbf{X}^T\mathbf{R}^T) - \mathbf{E}(\mathbf{R}\mathbf{X})\mathbf{E}(\mathbf{X}^T\mathbf{R}^T) \\
\iff \mathbf{C}' &= \mathbf{R}\mathbf{E}(\mathbf{X}\mathbf{X}^T)\mathbf{X}^T - \mathbf{R}\mathbf{E}(\mathbf{X})\mathbf{E}(\mathbf{X}^T)\mathbf{R}^T \\
\iff \mathbf{C}' &= \mathbf{R}(\mathbf{E}(\mathbf{X}\mathbf{X}^T) - \mathbf{E}(\mathbf{X})\mathbf{E}(\mathbf{X}^T))\mathbf{R}^T \\
\iff \mathbf{C}' &= \mathbf{R}\mathbf{C}\mathbf{R}^T
\end{aligned}
\tag{3.2}
$$

Where **C** is the covariance matrix, **C'** is the rotated covariance matrix, **R** is the rotation matrix, **E** is the expected value operand, **T** is the transpose operand, and **X** and **Y** are two jointly distributed real-valued random variables. With the IMU data, the angle ϕ between the map reference frame and the vehicle reference frame can be extracted and, thus, assembling the rotation matrix **R** with the following template matrix where ϕ is the angle between the map and the vehicle's reference frames.

$$
\begin{bmatrix}
\cos\phi & -\sin\phi & 0 \\
\sin\phi & \cos\phi & 0 \\
0 & 0 & 1
\end{bmatrix}
\tag{3.3}
$$

In building the pose graph, edges are added every time a new observation is made, connecting the observed features with the last odometry node recorded. When a node changes position (e.g., when a feature is considered moving), the previous edge is adjusted to match the node. This pose graph structure efficiently represents the vehicle's trajectories and the environment features.

Therefore, it is fundamental to understand the specific parameters of each detector and the importance of the correct data fusion between odometry and features data. These aspects contribute to assembling a more precise and reliable topological map, allowing autonomous navigation in complex environments.

3.3.3 Visualization

While the map is being built, it can be viewed in real-time through the RViz visualization tool. RViz is a component of ROS and allows for a real-time 3D view of several data structures commonly used in robotics, including sensor data, robot models, maps, and more. The map is represented with custom markers already integrated into RViz. RViz markers can be customized in color and scale to attribute different meanings to the represented object. The chosen basic custom markers are:

- **Green arrows** - odometry nodes.
- **Turquoise spheres** - landmark stationary nodes.
- **Purple spheres** - landmark moving nodes.
- **Yellow arrows** - represent odometry to odometry edges.

- **Orange arrows** - represent odometry to stationary landmark edges.
- **Red arrows** - represent odometry to moving landmark edges.
- **White Cylinders** - represent covariances.
- **White view oriented text** - node annotations.

To make the illusion of an ellipsis and, thus, accurately represent a covariance, the cylinder is oriented vertically with a small Z scale. In order to orient the ellipsis correctly, the X and Y-related values of the covariance matrix were used. The ellipsis vertexes and orientation angle can be calculated by assembling a new temporary matrix with these values. Its eigenvectors represent the principal axes or directions of a distribution's spread. The larger the corresponding eigenvalue, the greater the data spread along that eigenvector. The factor of the eigenvalues scaled the X and Y scales of the cylinder to achieve the ellipsis shape. The angle of the X component of the eigenvectors obtains the orientation angle of the ellipsis. Figure 3.10 shows an example of a covariance ellipsis and the other markers selected for the representation.

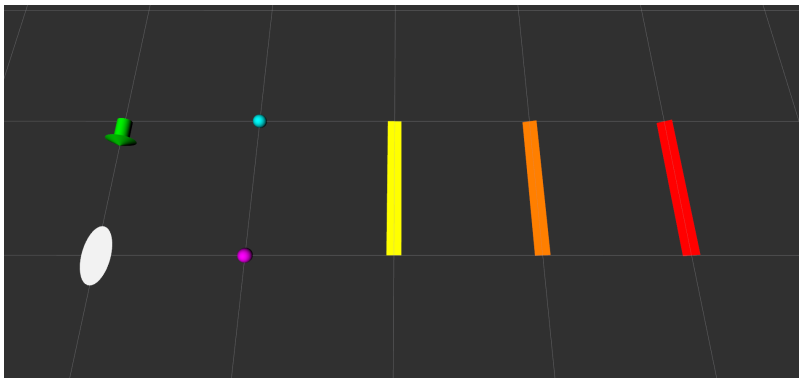


Figure 3.10: Example of each possible marker used for the visual scenario representation. The green arrow represents an odometry node, the shares represent landmark nodes (turquoise if stationary or purple if moving), the white ellipsis represents its associated node covariance and the lines connect the nodes (yellow connects odometry nodes, orange connects odometry nodes with stationary landmark nodes and red connects odometry nodes with moving landmarks).

Additionally, a data logger saves the raw sensor data each time a new edge is added, allowing for further analysis and the refinement of the mapping and navigation algorithms. This data is beneficial for inspection tasks because the data can be reviewed online.

This visual representation of the map elements allows a more transparent and intuitive understanding of the environment in real-time (while the map is being built) or offline (after the task is completed).

Chapter 4

Results

4.1 Scene perception and understanding

In the maritime scenario, multi-modality plays a vital role. It refers to integrating and utilizing multiple sensors to capture data from different modalities and sources. A multi-modality perception system allows for enhanced perception and understanding of the environment. In this context, using different detectors complements the perception of the environment, contributing to achieving multi-modality. In this regard, the detectors' precision influences the understanding and complementarity of the perception system.

As a result, the performance of the present solution's detectors was analyzed to determine the confidence level of the multi-modal perception system.

4.1.1 Dock detector

As mentioned in the previous chapter, the dock detector relies on LiDAR data and deep learning algorithms to identify and classify docks according to their type.

As described in Pereira's work [52], the detector follows a cascade architecture: docking structure detection and classification tasks. In both cases, detectors were trained under three levels of noise in the LiDAR points, generating three different neural networks for each case: trained with no noise, medium noise, and high noise. Similarly, the accuracy of the detector was measured under various noise levels in the LiDAR data, with a standard deviation of up to 1 meter. The noise was introduced in the Gazebo 3D simulator, where all the data was gathered.

The main metrics to evaluate this detector and classifier were accuracy, precision, and recall. For this binary classification task with a positive or negative label, the chosen confusion matrix is the following:

Recall reflects the total of true positives over actual positive instances in the dataset. A high recall indicates that the model has a lower tendency to miss a positive instance in binary classification. Contrary, a low recall reflects a higher number of false negatives in the inference process. The recall of the model can be calculated using the equation 4.1.

Table 4.1: Confusion matrix for a binary classification task.

		Predicted	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

$$Recall = \frac{TP}{TP+FN} \tag{4.1}$$

Precision is a metric that evaluates the accuracy of positive predictions, measuring how well the model avoids false negatives. A high-precision value indicates that the model reflects fewer false negative predictions. A low precision suggests that the model tends to produce false positives. Equation 4.2 shows how the precision can be calculated.

$$Precision = \frac{TP}{TP+FP} \tag{4.2}$$

Finally, accuracy measures the overall performance of the model. It reflects the ratio between the total correct predictions over the total number of instances on the dataset, as shown in equation 4.3. A high accuracy value indicates the model's effectiveness in the classification process, reflected by a high percentage of correct predictions. Conversely, a low accuracy value indicates low effectiveness and a low percentage of correct predictions.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{4.3}$$

Due to the cascade nature of the detector, two tests and validation processes were performed - one for each detector stage. In the first case, solely detection of docking structures, the summary of the results can be visualized in tables 4.2 and 4.3

Table 4.2: Accuracy of the detection stage.

Train Setup	Validation Accuracy	Test Accuracy
No Noise	96.16%	96.44%
Medium Noise	95.70%	95.99%
High Noise	94.45%	94.75%

Similarly, tables 4.4 and 4.5 reflect the classification task's performance under the same noise levels.

Table 4.3: Precision and accuracy of the detection stage.

		Test Setup			
		σ = 0.0m	σ = 0.2m	σ =0.5m	σ =1.0m
Train Setup	No Noise	P = 86.84%	P = 76.18%	P = 61.48%	P = 41.69%
		R = 86.67%	R = 63.73%	R = 27.01%	R = 18.97%
	Medium Noise	P = 86.75%	P = 86.52%	P = 82.44%	P = 58.61%
		R = 86.27%	R = 85.36%	R = 79.63%	R = 42.09%
	High Noise	P = 86.97%	P = 86.13%	P = 83.42%	P = 73.87%
		R = 85.96%	R = 85.19%	R = 81.75%	R = 71.03%

4.1.2 Vessel detector

As referenced in the previous chapter, the vessel detector relies on camera information to detect and classify the objects present in the maritime scenario. The detector performs its task by feeding a custom-trained YOLO-based network, as described in chapter 3.

The classifier's predictions for each class are described in the confusion matrix. The counts for true positives, false positives, true negatives, and false negatives are summarized. Figure 4.1 shows a diagram of the classifier's confusion matrix. As observed, the classifier can accurately distinguish the different types of vessels.

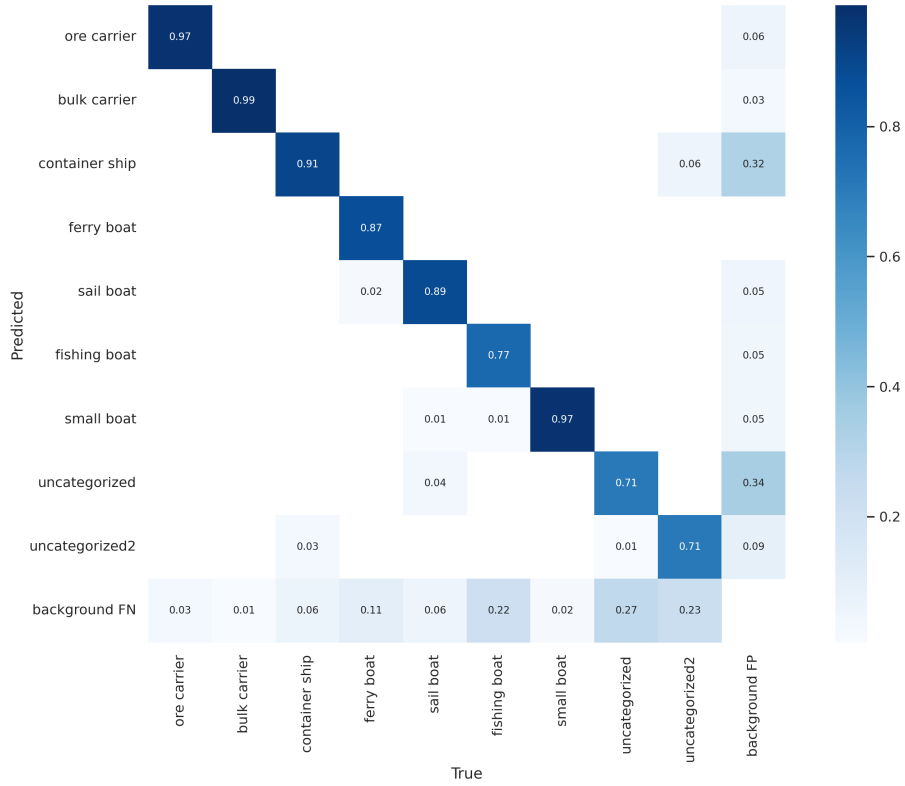


Figure 4.1: Custom trained vessel detector's confusion matrix.

The F1 score measures the classifier's accuracy by combining. the precision and recall metrics,

Table 4.4: Accuracy of the detection stage.

Train Setup	Validation Accuracy	Test Accuracy
No Noise	86.90%	86.70%
Medium Noise	86.15%	86.16%
High Noise	85.25%	85.86%

contrary to the accuracy metric, were calculated based on how many correct predictions happened across the test batch. The F1 score balances the precision and recall at different threshold values, showcasing these two metrics' trade-offs. The F1 score can be calculated by:

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{4.4}$$

The F1 curve shown in figure 4.2 translates into overall good results. The uncategorized classes can be considered outliers because they are not vessel classes and are more heterogeneous among their occurrences in the dataset.

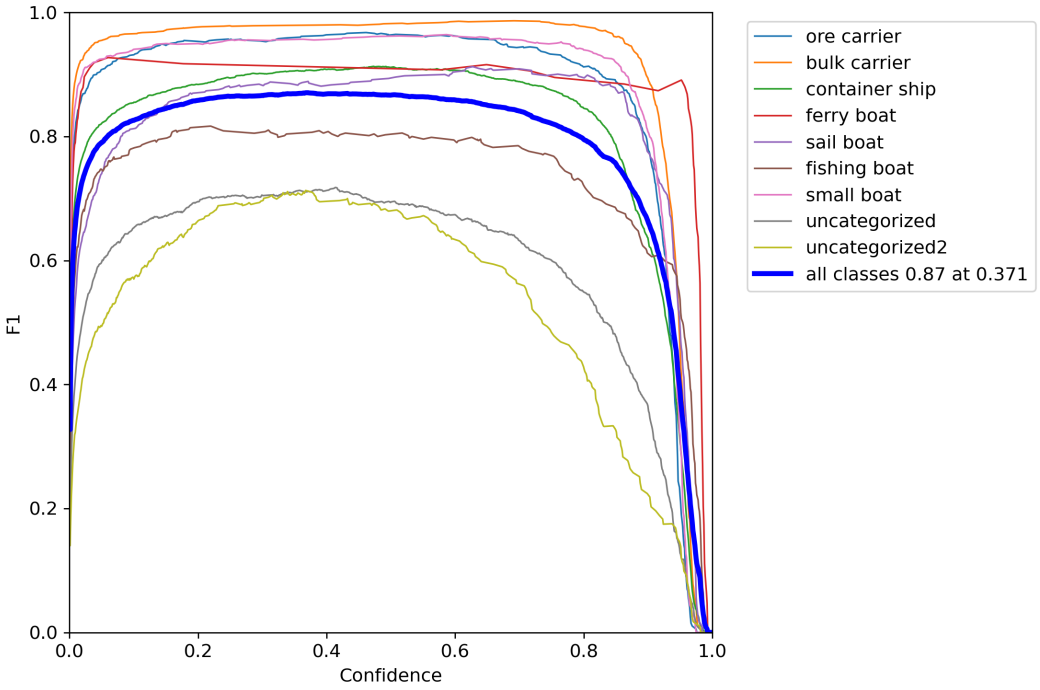


Figure 4.2: Custom trained vessel detector's F1 curve.

The precision-recall curve shown in image 4.3 illustrates the relationship between these two metrics at different threshold values, reflecting the imbalance of the dataset. This curve helps in the decision of the threshold value. As seen in the graph, the detector performed well, apart from the two outlier classes, allowing the selection of a high threshold value in the detection and

Table 4.5: Precision and accuracy of the classification stage.

		Test Setup			
		σ = 0.0m	σ = 0.2m	σ =0.5m	σ=1.0m
Train Setup	No Noise	P = 96.48%	P = 92.97%	P = 74.85%	P = 68.62%
		R = 96.44%	R = 92.56%	R = 65.14%	R = 58.88%
	Medium Noise	P = 96.04%	P = 94.35%	P = 89.36%	P = 76.15%
		R = 95.99%	R = 94.21%	R = 87.88%	R = 65.55%
	High Noise	P = 94.96	P = 91.20%	P = 88.72%	P = 87.76%
		R = 94.75%	R = 91.19%	R = 88.71%	R = 87.63%

classification task. By this curve and the F1 curve, it is also possible to conclude that the *fishing boat* class is underrepresented in the dataset.

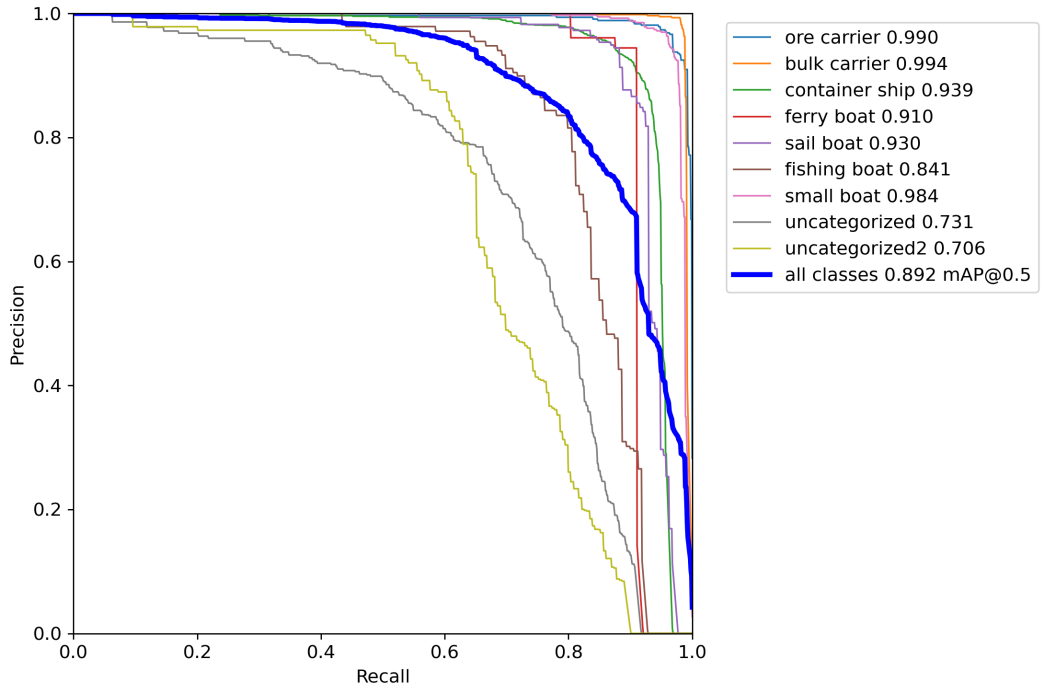


Figure 4.3: Custom trained vessel detector's precision-recall curve.

4.2 Scenario representation

As discussed in chapter 3, the scenario is represented by a pose graph. To assess the reliability and resemblance with reality, several tests with increasing difficulty and complexity were performed in simulated and real environments. The test scenarios include:

1. Manually controlled data input;
2. Simple scenario in a simulated world;

3. Complex scenario with dynamic objects in a simulated world;

4. Real scenario in the ATLANTIS test center.

The overlap of a number of point clouds from a sliding window was used as the ground truth for the evaluation of the pose. In the simulated scenarios, measurements from the pose graph point to the object it represents were performed to evaluate the map's accuracy metrically. This operation was achieved with the RVIZ tool. When this operation was not possible, only qualitative analysis was performed.

The used detectors and the provided odometry systems do not output a covariance matrix. Therefore, an arbitrary covariance value was used to test its representation in the topological map.

4.2.1 Manually controlled data inputs

In order to isolate from noise and/or possible problems with the feature detectors, data was manually fed into the map builder algorithm.

This scenario contains two stationary obstacles, one moving obstacle, and a trajectory describing an 'L' shape, represented by figure 4.4. Due to the scale of the map's visual representation, it is possible to read the vertex numbers corresponding to each node. These numbers are assigned incrementally as each node is added to the map and kept equal, even if their respective nodes are moved for pose corrections. The moving obstacle, represented by vertex number six, moved along the negative direction of the Y axis, represented in green on the frame origin. This obstacle was only visible once the vehicle started performing a curve at vertex five. This test simulated that vertex four could not be successfully detectable after vertex seven (including), as represented by the absence of an edge connecting the following odometry nodes to that landmark, as observed in the map. The map's scale equals one meter per square in the displayed grid.

4.2.2 Simple scenario in a simulated world

In this scenario, three types of stationary docks exist in a simulated environment, represented in figure 4.5a. The odometry and map frames are coincident, and the ASV starts approximately in their origin position.

The resulting map can be visualized in figure 4.5b. The ASV performs an S-shaped trajectory between the T-shaped and U-shaped dock represented by the yellow lines. As it can be visually confirmed, the feature nodes referring to docks overlap with the ground truth, confirming the qualitative accuracy of this representation.

In this simple scenario, all docks are detectable when the algorithm starts to map the area. However, as observed on the map, sometimes the detector loses one or more docks. Once the algorithm re-detects the same dock, it assigns the detection to the old dock instead of adding a new node.

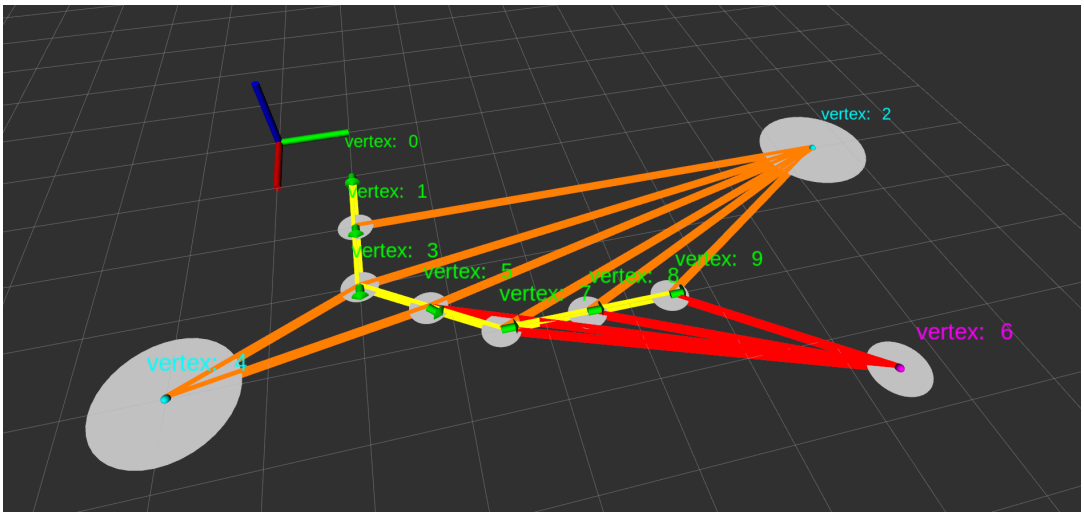


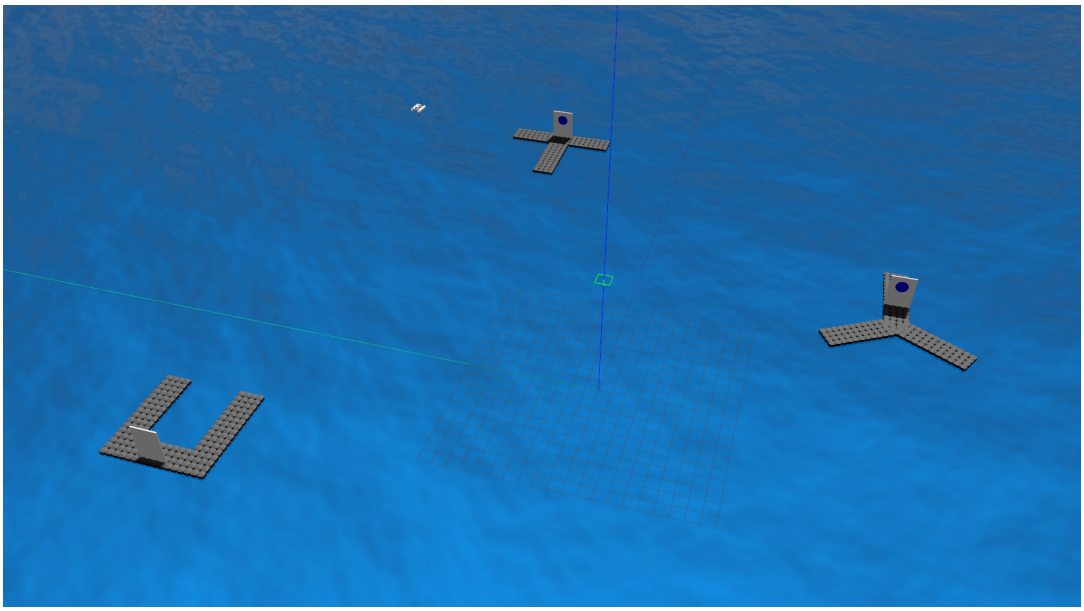
Figure 4.4: Map's visual representation resulting from manually feeding odometry points and features into the map builder algorithm. Vertices two and four are stationary landmarks, vertex six is a moving obstacle, and vertices zero, one, three, five, seven, eight, and nine are odometry nodes.

4.2.3 Complex scenario with dynamic objects in a simulated world

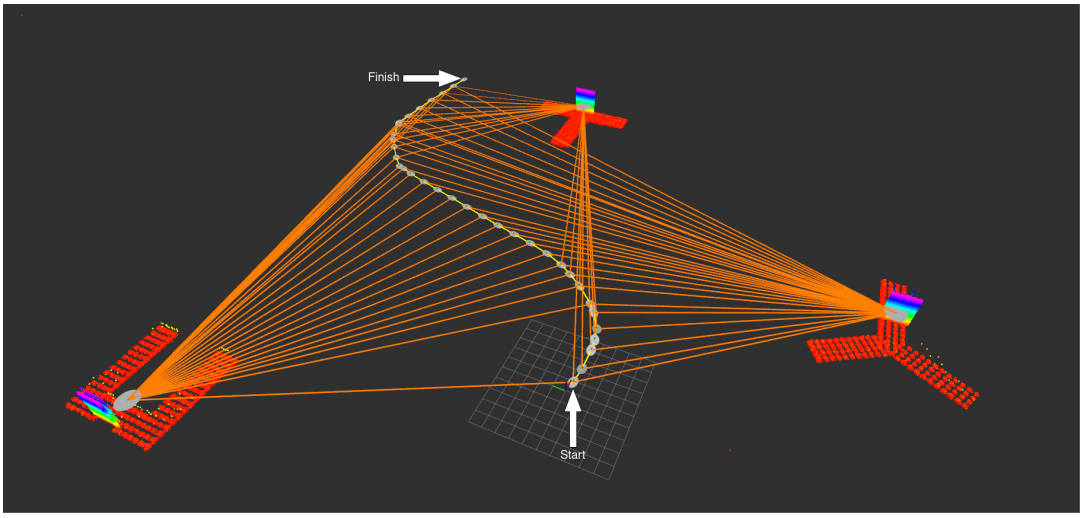
This scenario is composed of three stationary docks and a moving ship in the configuration shown in figure 4.6a. At the start of the scene, the ship occludes the U-shaped dock and moves with the trajectory represented by the green arrow while the SENSE ASV navigates around the ship, thus revealing the dock.

As expected, the vessel was detected and represented on the map until it went out of the field of view of the ASV's camera. The point where all the red edges converge is the last detection recorded. Once the ship revealed the U-shaped dock, it became detectable and was successfully represented on the map.

However, the back side of the ship resembles a U-shape dock due to its shape, therefore it was detected as a dock and it is represented on the map and it is pointed by the pink arrow in figure 4.6b.

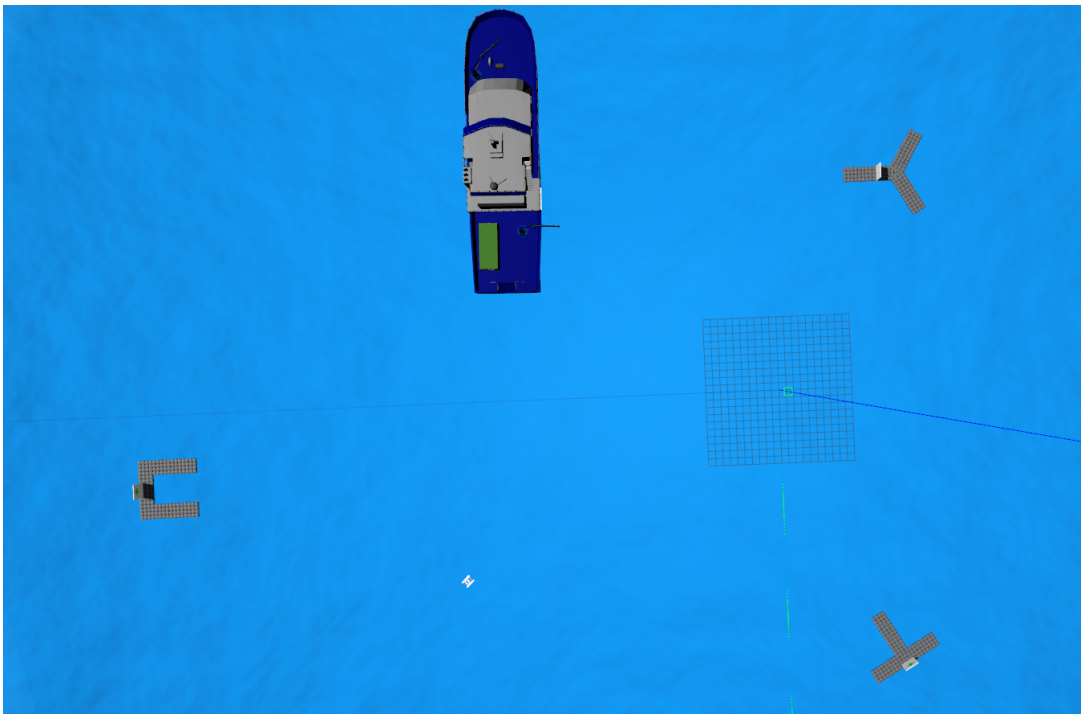


(a) Simple scenario on a simulated environment.

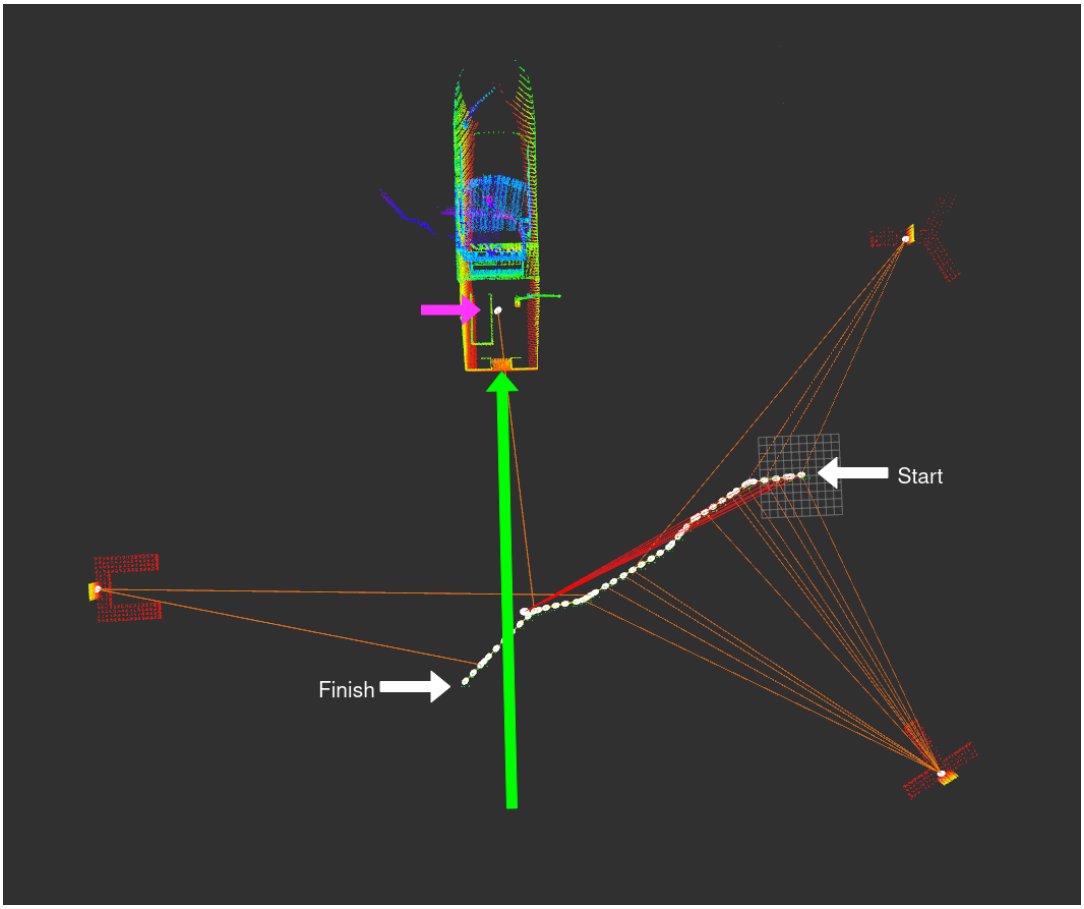


(b) Resulting topological map and respective ground truth.

Figure 4.5: Simulated environment test in a simple scenario - three docks are positioned in a triangle configuration, and the ASV navigates with an S-shape trajectory between the U-shaped and T-shaped docks.



(a) Complex scenario with a moving ship on a simulated environment.



(b) Resulting topological map and respective ground truth.

Figure 4.6: Complex scenario with dynamic obstacles test - the static objects are the same as for the previous test. The dynamic object consists of a large boat that crosses the robot's path, including one of the docks at the beginning of the test.

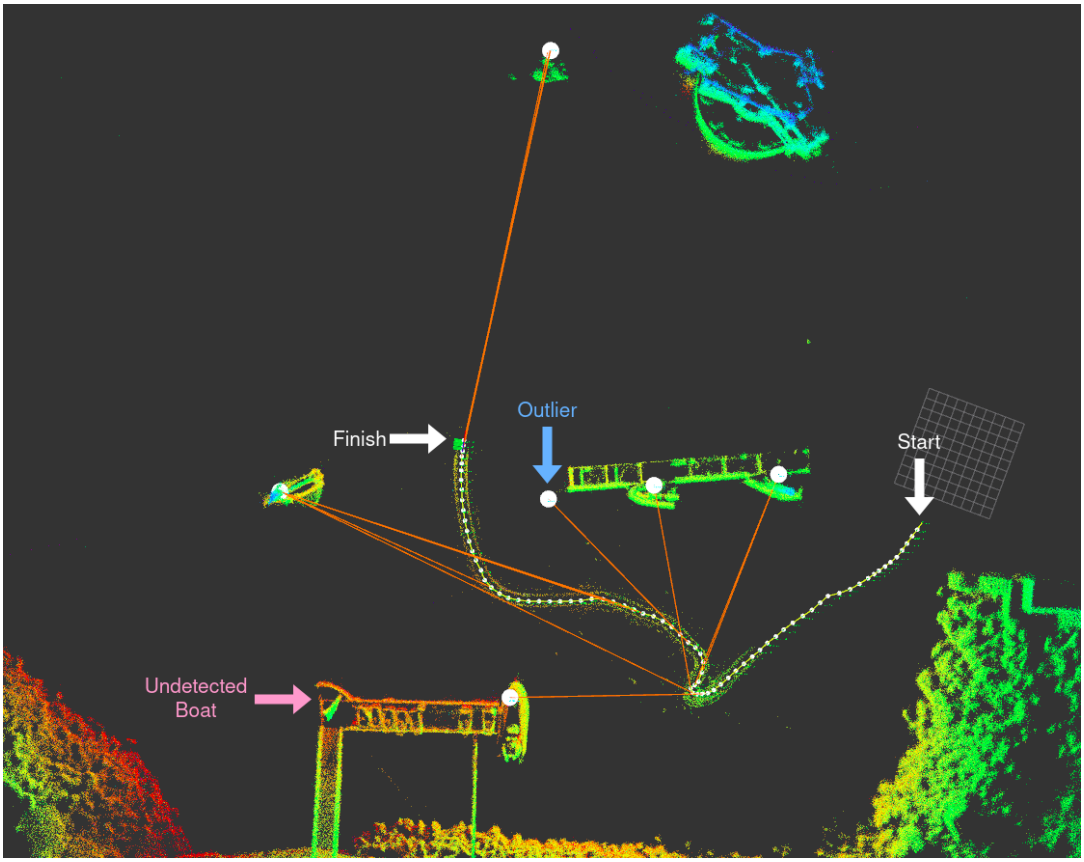
4.2.4 Real scenario in the ATLANTIS test center

In this scenario, several small boats were parked in the ATLANTIS test center in the configuration shown in image 4.7a and its respective map in figure 4.7b. The Nautilus ASV (red ellipsis in figure 4.7a) navigated around the central dock (yellow ellipsis), remotely controlled. Between the elapsed time between figure in 4.7b and 4.7a, the vessel in the blue circle in figure 4.7a left the area, and a sailboat docked in the area defined by the green circle.

The Nautilus ASV navigated with the trajectory shown in figure 4.7b. During the navigation, the vessel marked by the pink arrow in figure 4.7b was never inside the field of view of the ASV because of an obstruction in the line of sight between the ASV and that said vessel. Therefore, it could not be detected and added to the map. In the generated map, an outlier is present and marked by the blue arrow in figure 4.7b. A plausible explanation for this occurrence is the temporal desynchronization between the classified image and the point cloud. This asynchronism is caused by a long inference time in the classification process, resulting in the depth estimation of a point from a past image using a point cloud from the present time. In this case, the discrepancy between the correct pose (already represented in the map) and the new inaccurate estimate was enough for the algorithm to assume a new vessel in the area.

It is relevant to note that there are fewer feature detections in this test scenario. This happened because the vessel detector could not detect the vessels whenever they were within the ASV's field of view. High exposure in the collected images or the lack of enough points collected by the LiDAR to estimate the pose of the vessel confidently could justify the low number of detections throughout the test scenario.

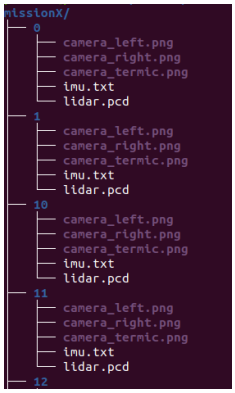
(a) Aereal view of the ATLANTIS test center's configuration.



(b) Resulting topological map and respective ground truth.

Figure 4.7: Real scenario test - six vessels are docked in the ATLANTIS test center. The ASV navigates around the central dock, represented by the yellow ellipsis.

The data logger was enabled during this test, and data was captured for later analysis. The logger successfully recorded all 63 observations. Although there are less than 63 edges on the map, many were moved to accommodate node corrections, but the sensor data is still collected. The logger creates a folder in a specified location and saves the data in a tree structure, where each sub-folder is named after the observance order. Figure 4.8a shows part of the generated file tree. In this example, sonar was not available for recording. The images retrieved by the three onboard cameras (left, right, and thermal) are saved in PNG format. The point cloud frames retrieved by the sonar and LiDAR are saved in PCD format. Finally, GPS and IMU data are saved in plain text in TXT format. GPS data saves the header (containing a timestamp, sequence number, and reference frame ID), altitude, and the GPS coordinates - longitude and latitude. The IMU file contains a header and the quaternion that represents the rotation of the vehicle in that time instance. Figure 4.8b shows an example of a randomly picked observation from the left camera.



(a) Generated file tree.

(b) Example of an automated collected image by the logger.

Figure 4.8: Data collected by the logger for posterior analysis.

4.2.5 Performance Analysis

The time difference between the instant when a sensor data sample is collected and the moment when it is processed into a feature was measured to determine the time performance of the algorithm in conjunction with the detectors. This interval was determined by manually feeding each detector with its respective data sample, measuring the time instant of the clock, and measuring it again once the algorithm successfully returned the feature data structure. By feeding a point cloud with docks present in the surroundings of the ASV, it took an average of 148,8ms. An image fed to the vessel detector, along with the depth estimator, resulted in an average process time of 216.7ms.

Chapter 5

Conclusions and future work

This dissertation proposes a method to semantically represent environments, with the ultimate goal of enabling safer navigation through future implementation of obstacle avoidance. The developed system can handle multiple feature detectors and combine them into a single representation by standardizing the data structure output by each detector module. Data is then represented in a pose graph-based topological map and can be visualized in real-time and saved for posterior analysis. The suggested architecture, where each detector's data is standardized into a uniform data structure, allows the developed system to be easily scalable so it can be deployed in multiple vehicles - in this case, the Nautilus ASV and the SENSE ASV.

Initially, the algorithm was tested in a simulated environment, where mostly docks were present. Since this detector had high precision, accuracy, and recall, the generated map reliably represented the scenario. The detectors' quality proved fundamental for an accurate topological representation. As seen in the third test - complex scenario with dynamic objects in a simulated world, a false positive detection negatively impacts the overall representation. Therefore, using good detectors is fundamental to achieving a good representation of the environment. Moreover, identifying and removing outliers in the map could improve the fidelity of the representation. In the real test scenario, vessels took the majority of the features that could be detected. This detector proved not accurate enough for the task because it could not detect and estimate the distance of all vessels whenever they were within the sensor's field of view. Another possible reason for the errors in the pose estimation is the long inference time to classify the vessels causing data to be out of sync with the rest of the sensors and, thus, failing to estimate the vessel's distance to the ASV. The false positive that can be seen in the finished map derives from this circumstance because the logs revealed that the outlier feature is the vessel that is near it and correctly identified. To solve this issue, two solutions can be implemented. The first is to change to YOLOv8 [58], which promises better performance than the used version. Additionally, the use of a stereo pair of cameras could benefit the estimation of the distance between camera-detectable features and the ASV.

Concluding, the developed work is able to represent real scenarios in topological maps, but its accuracy is tied to the detectors' accuracy and precision. Besides the previously mentioned

improvement proposed, further development can be done, such as:

- Represent moving landmarks in the visualizer;
- Use this methodology to feed an AVOA algorithm to perform obstacle avoidance;
- Implement simultaneous mapping and location (SLAM) algorithm based this topological map implementation.

References

[1] BA Neil Baird. Fatal ferry accidents, their causes, and how to prevent them. 2018.

[2] Daniel Filipe Campos, Anibal Matos, and Andry Maykol Pinto. Modular multi-domain aware autonomous surface vehicle for inspection. *IEEE Access*, 10:113355–113375, 2022.

[3] Bing Qiang Huang, Guang Yi Cao, and Min Guo. Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance. *2005 International Conference on Machine Learning and Cybernetics, ICMLC 2005*, pages 85–89, 2005.

[4] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. *IEEE International Conference on Intelligent Robots and Systems*, 2016-November:2759–2764, 11 2016.

[5] Chirayu Garg. Stereo-visual-odometry, 2018.

[6] Daniel Filipe Campos, Anibal Matos, and Andry Maykol Pinto. An adaptive velocity obstacle avoidance algorithm for autonomous surface vehicles. *IEEE International Conference on Intelligent Robots and Systems*, pages 8089–8096, 11 2019.

[7] Tae Hyeon Nam, Jae Hong Shim, and Young Im Cho. A 2.5d map-based mobile robot localization via cooperation of aerial and ground robots. *Sensors (Switzerland)*, 17, 12 2017.

[8] W. Roque and D. Doering. Constructing approximate voronoi diagrams from digital images of generalized polygons and circular objects. 2003.

[9] Boliang Guan, Shujin Lin, Ruomei Wang, Fan Zhou, Xiaonan Luo, and Yongchuan Zheng. Voxel-based quadrilateral mesh generation from point cloud. *Multimedia Tools and Applications*, 79:20561–20578, 8 2020.

[10] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. 7 2022.

[11] Andry Maykol Pinto, Joao V. Amorim Marques, Daniel Filipe Campos, Nuno Abreu, Anibal Matos, Martio Jussi, Robin Berglund, Jari Halme, Petri Tikka, Joao Formiga, Christian Verrecchia, Serena Langiano, Clara Santos, Nuno Sa, Jaap-Jan Stoker, Fabrice Calderoni, Shashank Govindaraj, Alexandru But, Leslie Gale, David Ribas, Natalia Hurtos, Eduard Vidal, Pere Ridao, Patryk Chieslak, Narcis Palomeras, Stefano Barberis, and Luca Aceto. Atlantis - the atlantic testing platform for maritime robotics. pages 1–5. IEEE, 9 2021.

[12] Dony Hutabarat, Muhammad Rivai, Djoko Purwanto, and Harjuno Hutomo. Lidar-based obstacle avoidance for the autonomous mobile robot. *Proceedings of 2019 International Conference on Information and Communication Technology and Systems, ICTS 2019*, pages 197–202, 7 2019.

[13] Jiechao Liu, Paramsothy Jayakumar, Jeffrey L. Stein, and Tulga Ersal. A study on model fidelity for model predictive control-based obstacle avoidance in high-speed autonomous ground vehicles. *http://dx.doi.org/10.1080/00423114.2016.1223863*, 54:1629–1650, 11 2016.

[14] Aggrey Shitsukane, W Cheriuyot, C Otieno, and Mvurya Mgala. A survey on obstacles avoidance mobile robot in static unknown environment. *International Journal of Computer (IJC)*, 12 2018.

[15] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 500–505, 1985.

[16] Fethi Matoui, Boumedyen Boussaid, and Mohamed Naceur Abdelkrim. Local minimum solution for the potential field method in multiple robot motion planning task. *16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, STA 2015*, pages 452–457, 7 2016.

[17] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1:28–39, 11 2006.

[18] Darrall Henderson, Sheldon H. Jacobson, and Alan W. Johnson. The theory and practice of simulated annealing. *Handbook of Metaheuristics*, pages 287–319, 2 2006.

[19] P. L. Bonate. A brief introduction to monte carlo simulation. *Clinical Pharmacokinetics*, 40:15–22, 2001.

[20] Yi Zhang, Ping Sun, Yuhan Yin, Lin Lin, and Xuesong Wang. Human-like autonomous vehicle speed control by deep reinforcement learning with double q-learning. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2018-June:1251–1256, 10 2018.

[21] Daniel N. Cassenti, Vladislav D. Veksler, and Frank E. Ritter. Editor's review and introduction: Cognition-inspired artificial intelligence. *Topics in Cognitive Science*, 14:652–664, 10 2022.

[22] Siyuan Chen and Ken Mai. Towards specialized hardware for learning-based visual odometry on the edge. *IEEE International Conference on Intelligent Robots and Systems*, 2022-October:10603–10610, 2022.

[23] David Fernandez and Andrew Price. Visual odometry for an outdoor mobile robot. *2004 IEEE Conference on Robotics, Automation and Mechatronics*, pages 816–821, 2004.

[24] Kurt Konolige, Motilal Agrawal, and Joan Solà. Large-scale visual odometry for rough terrain. *Springer Tracts in Advanced Robotics*, 66:201–212, 2010.

[25] David Valiente García, Lorenzo Fernández Rojo, Arturo Gil Aparicio, Luis Payá Castelló, and Oscar Reinoso García. Visual odometry through appearance- and feature-based method with omnidirectional images. *Journal of Robotics*, 2012:1–13, 2012.

[26] Tommi Tykkälä, Cédric Audras, and Andrew I. Comport. Direct iterative closest point for real-time visual odometry. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2050–2056, 2011.

[27] Kwanghae Heo and Dong Hoon Lim. Noise reduction using patch-based cnn in images. *The Korean Data I& Information Science Society*, 30:349–363, 3 2019.

[28] Changhao Chen, Stefano Rosa, Yishu Miao, Chris Xiaoxuan Lu, Wei Wu, Andrew Markham, and Niki Trigoni. Selective sensor fusion for neural visual-inertial odometry, 2019.

[29] Woong Gie Han, Seung Min Baek, and Tae Yong Kuc. Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 3:2747–2751, 1997.

[30] Zoltán Gyenes, Ladislau Bölöni, and Emese Gincsainé Szádeczky-Kardoss. Can genetic algorithms be used for real-time obstacle avoidance for lidar-equipped mobile robots? *Sensors 2023, Vol. 23, Page 3039*, 23:3039, 3 2023.

[31] Adem Tuncer and Mehmet Yildirim. Dynamic path planning of mobile robots with improved genetic algorithm. *Computers I& Electrical Engineering*, 38:1564–1572, 11 2012.

[32] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 4 2002.

[33] K. Kamil, K. H. Chong, H. Hashim, and S. A. Shaaya. A multiple mitosis genetic algorithm. *IAES International Journal of Artificial Intelligence*, 8:252–258, 2019.

[34] Hanqi Wang, Zhiling Wang, Linglong Lin, Fengyu Xu, Jie Yu, Huawei Liang, Citation : Wang, H ; Wang, Z ; Lin, L ; Xu, F ; Yu, J ; Liang, and H Optimal. Optimal vehicle pose estimation network based on time series and spatial tightness with 3d lidars. *Remote Sensing 2021, Vol. 13, Page 4123*, 13:4123, 10 2021.

[35] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 6 1965.

[36] Aggrey Shitsukane, Wilson Cheruiyot, Calvin Otieno, and Mgala Mvurya. Fuzzy logic sensor fusion for obstacle avoidance mobile robot. In *2018 IST-Africa Week Conference (IST-Africa)*, pages Page 1 of 8–Page 8 of 8, 2018.

[37] Anish Pandey, Rakesh Kumar Sonkar, Krishna Kant Pandey, and D. R. Parhi. Path planning navigation of mobile robot with obstacles avoidance using fuzzy logic controller. *2014 IEEE 8th International Conference on Intelligent Systems and Control: Green Challenges and Smart Solutions, ISCO 2014 - Proceedings*, pages 36–41, 5 2014.

[38] Cang Ye, Nelson H.C. Yung, and Danwei Wang. A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33:17–27, 1 2003.

[39] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm Intelligence 2007 1:1*, 1:33–57, 8 2007.

[40] Christoph Rosmann, Frank Hoffmann, and Torsten Bertram. Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control. *2015 European Control Conference, ECC 2015*, pages 3352–3357, 11 2015.

[41] Alireza Asvadi, Paulo Peixoto, and Urbano Nunes. Detection and tracking of moving objects using 2.5d motion grids. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2015-October:788–793, 10 2015.

[42] David Thompson, Eric Coyle, and Jeremy Brown. Efficient lidar-based object segmentation and mapping for maritime environments. *IEEE Journal of Oceanic Engineering*, 44:352–362, 4 2019.

[43] Sang Il Oh and Hang Bong Kang. Fast occupancy grid filtering using grid cell clusters from lidar and stereo vision sensor data. *IEEE Sensors Journal*, 16:7258–7266, 10 2016.

[44] Kamel Mekhnacha, Yong Mao, David Raulo, Christian Laugier, and Christian Laugier Bayesian. Bayesian occupancy filter based "fast clustering-tracking" algorithm. 9 2008.

[45] Longin Jan Latecki and Rolf Lakaemper. Polygonal approximation of laser range data based on perceptual grouping and em. *Proceedings - IEEE International Conference on Robotics and Automation*, 2006:790–796, 2006.

[46] Benchun Zhou, Aibo Wang, Jan Felix Klein, and Furmans Kai. Object detection and mapping with bounding box constraints. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2021.

[47] Juntao Yang and Zhizhong Kang. Voxel-based extraction of transmission lines from airborne lidar point cloud data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11:3892–3904, 10 2018.

[48] Yusheng Xu, Xiaohua Tong, and Uwe Stilla. Voxel-based representation of 3d point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*, 126:103675, 6 2021.

[49] Dhiego Bersan, Renato Martins, Mario Campos, and Erickson R. Nascimento. Semantic map augmentation for robot navigation: A learning approach based on visual and depth data. *Proceedings - 15th Latin American Robotics Symposium, 6th Brazilian Robotics Symposium and 9th Workshop on Robotics in Education, LARS/SBR/WRE 2018*, pages 51–57, 12 2018.

[50] André Silva Aguiar, Filipe Neves Dos Santos, José Boaventura Cunha, Héber Sobreira, and Armando Jorge Sousa. Localization and mapping for robots in agriculture and forestry: A survey. *Robotics 2020, Vol. 9, Page 97*, 9:97, 11 2020.

[51] Zhike Zhang, Tianle Wang, Shuxia Gu, and Zhiyu Xiang. Mpf: A robust vehicle localization framework based on topological map and odometry. *IEEE Sensors Journal*, 4 2023.

[52] Maria Inês, Rodrigues Pereira, Pedro Nuno, and Barbosa Leite. A machine learning approach for predicting docking-based structures. 7 2020.

[53] Diogo Ferreira Duarte, Maria Ines Pereira, and Andry Maykol Pinto. Multiple vessel detection and tracking in harsh maritime environments. pages 1–5. IEEE, 9 2021.

[54] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics 2020, Vol. 9, Page 1295*, 9:1295, 8 2020.

[55] Y. M. Wang, Y. Li, and J. B. Zheng. A camera calibration technique based on opencv. In *The 3rd International Conference on Information Sciences and Interaction Sciences*, pages 403–406, 2010.

[56] Ankit Dhall, Kunal Chelani, Vishnu Radhakrishnan, K Madhava Krishna, A Dhall, K Chelani, V Radhakrishnan, and KM Krishna. Lidar-camera calibration using 3d-3d point correspondences. 5 2017.

[57] Andry Maykol Pinto, Paulo G. Costa, Miguel V. Correia, Anibal C. Matos, and A. Paulo Moreira. Visual motion perception for mobile robots through dense optical flow fields. *Robotics and Autonomous Systems*, 87:1–14, 1 2017.

[58] G. Jocher, A. Chaurasia, and J. Qiu. Yolo by ultralytics, 2023.