Towards improving the perception and autonomous decision making in underwater dynamic docking

Hassan Elkholy

Master Thesis

Erasmus Mundus Master in
Marine and Maritime Intelligent Robotics

Universitat Jaume I

July 19, 2023

Supervised by: Alaa El Jawad and Prof.Antonio Morales

Co-funded by the
Erasmus+ Programme
of the European Union

B

Acknowledgments

First of all, I would like to thank my Master Thesis supervisor, Alaa ElJawad, Robotics lead, Forssea Robotics for his tremendous support during my internship at Forssea Robotics. His feedback was helpful and vital to the development of this work. I also would like to thank Jaoaud Hajjami, Forssea Robotics for his feedback and cooperation in the AI and vision parts of this work. I would also to thank Ian Macelory and Auguste Bourgois, Forssea Robotics for their help and consults during my internship. I would like also to thank a lot Prof.Antonio Morales, my academic supervisor from Jaume I University (UJI) for his continuous support and feedback regarding the development of this work. His feedback was essential and important in developing this work. I would like also to thank Prof.Pedro Sanz, MIR master coordinator at UJI for his continuous support regarding the process of working on the master thesis as well as solving all the administrative obstacles. Least but not the least, I thank all the colleagues at Forssea Robotics for their continuous support and the friendly work environment!

Abstract

Nowadays, there is a growing need for using marine robotics for a variety of tasks such as inspection, maintenance, and repair, commonly known as IMR operations. One of the main challenges associated with using such vehicles is retrieving them. The retrieval process can be solved by docking the vehicle into a docking station. There are primarily two types of docking stations: stationary and floating. There have been several advancements in docking into stationary stations which is relatively easier than docking into floating stations. On the other hand, there is very little work in the literature regarding docking into floating stations. Docking into floating stations is needed in the case of retrieving remotely operated vehicles (ROVs), which can be done by the surface vessel. Depending on the sea state and visibility conditions, this task can be quite challenging, even for the most experienced pilots. That's why there is research to enable autonomous docking for floating docking station scenarios. This report will discuss several approaches to improve perception and autonomous decision-making in dynamic autonomous docking for ROVs. The work done here is part of my internship at Forssea Robotics company. The proposed solutions include building a simulation environment for underwater dynamic docking as there are no available simulations for such scenarios. The solutions also include several approaches for estimating the position of the docking garage as well as algorithms for detecting the garage using AI. And finally, an autonomous control strategy will be discussed.

Contents

Contents **v**

1 Introduction **1**
1.1 Work Motivation 1
1.2 Objectives 2
1.3 Environment and Initial State 4
1.4 Contributions 4
1.5 Outline of the thesis 5

2 Planning and resources evaluation **7**
2.1 Planning 7
2.2 Resource Evaluation 7

3 Literature review **9**
3.1 Underwater autonomous docking 9
3.2 Inspirations from other fields 13
3.3 Control strategies for underwater autonomous docking 15

4 System Analysis and Design **19**
4.1 Docking scenario and challenges 19
4.2 System overview 22
4.3 Software and tools used 24

5 Work Development **29**
5.1 Building the simulation 29
5.2 Homing - Garage detection 36
5.3 Homing - Control strategy 39
5.4 Docking - Motion estimation 41
5.5 Collecting datasets from the real garage 48

6 Results **51**
6.1 Gazebo simulation of docking garage 51
6.2 Motion prediction 58
6.3 Garage detection 64

6.4 Collecting real dataset 65

7 Conclusions and Future Work **67**
7.1 Conclusions 67
7.2 Future work 68

Bibliography **69**

A Other considerations **71**
A.1 Appendices 71

CHAPTER 1

INTRODUCTION

Without goals and plans to reach them, you are like a ship that has set sail with no destination - Fitzhugh Dodson

Contents

1.1	Work Motivation	**1**
1.2	Objectives	**2**
1.3	Environment and Initial State	**4**
1.4	Contributions	**4**
1.5	Outline of the thesis	**5**

This chapter will provide a brief introduction about the motivation behind my desire to work in this area in general as well as the motivation behind my internship in particular. It will also demonstrate the main objectives of my internship as well as the environment and initial state.

1.1 Work Motivation

Inspection, Maintenance, and Repair (IMR) missions are critical operations in the offshore industry to ensure the safe and effective functioning of offshore assets such as oil rigs, wind farms, pipelines, etc. These missions involve equipment inspection and testing, routine maintenance to prevent failures, and repairs to correct any flaws or damage that may occur. Such operations are required to maintain assets operating at peak efficiency and prevent accidents in the harsh underwater environment. The offshore industry is a major player in the global economy in general and the blue economy in particular. One of the approaches to conducting IMR missions is by utilizing Remotely Operated Vehicles

(ROVs) and Autonomous Underwater Vehicles (AUVs) due to their unique capabilities and numerous advantages. These robotic systems are deployed to perform critical tasks in challenging harsh underwater environments and ensure the efficient and cost-effective management of offshore assets. Inspection is one of the primary applications of ROVs and AUVs in offshore IMR missions. Equipped with high-resolution cameras, sensors, and imaging technologies, these vehicles are capable of conducting detailed visual inspections of underwater structures, pipelines, and equipment. They can also assist in conducting proactive maintenance planning and early defect identification. Moreover, one of the challenges associated with using these vehicles is vehicle retrieval. Usually, this process is conducted by an experienced pilot where the pilot will try to dock the vehicle to a floating docking garage after the mission. To further improve the operation and reduce the costs and the need for experienced pilots, the research community has been recently focusing on the problem of autonomous underwater docking. Implementing efficient autonomous underwater docking will reduce operational costs significantly as well as increase the autonomous capabilities of the conducted missions. However, developing a robust autonomous underwater docking system has several challenges such as bad perception, the slow dynamics of the vehicle, the risk of damaging the vehicle of the garage, and of course, the autonomous decision-making capability. There are a few companies that work on issues related to autonomous underwater docking; one of them is Forssea Robotics company. The main motivation for my internship at Forssea was to work on the underwater autonomous docking problem by achieving the objectives explained in section 1.2.

1.2 Objectives

There are several objectives to my master thesis internship at Forssea Robotics company. All the objectives fall under the big umbrella of advancing the perception and autonomous decision-making in autonomous underwater docking. These objectives can be summarized as the following:

Objective 1: Conducting State-of-the-Art Research on Underwater Autonomous Docking Solutions and Challenges

My main objective for my master's thesis internship is to conduct comprehensive state-of-the-art research on underwater autonomous docking solutions. One of my tasks was to study and analyze existing literature, research papers, and relevant technologies and come up with summaries and suggestions to discuss later with the rest of the team. Through this research, my objective was to gain insights into the various challenges faced by autonomous docking systems and identify potential areas for improvement and possible solutions.

Objective 2: Developing a High-Quality Simulation Environment using Gazebo

A crucial aspect of my internship involves creating a realistic simulation environment for the docking scenario using the Gazebo simulator. Given the absence of a simulation specifically designed for the docking scenario and the docking garage, it was my task to utilize the up-to-date capabilities of Gazebo to construct a realistic and accurate simulation model. By simulating the real behaviour of the docking garage, this simulation will serve as a valuable tool for testing and validating different autonomous docking algorithms and strategies. A challenge associated with this objective was the lack of information about the real behaviour of the docking garage in the literature. One of the very few resources I could find was the work and experiments by the Centre for Robotics and Intelligent Systems (CRIS), University of Limerick in the North Atlantic Ocean [12] [20] [18].

Objective 3: Implementing a Proof of Concept (POC) for Autonomous Docking

Building upon the simulation environment, my next objective was developing and implementing a good proof of concept (POC) solution for improving autonomous docking. Based on my research findings and the insights gained from the simulation, I was asked to propose a novel approach or algorithm that addresses the existing challenges in underwater autonomous docking. This POC implementation will demonstrate the feasibility and effectiveness of my proposed solution within the simulated environment that I already developed, showcasing its potential for real-world application. In general, conducting a POC is the first step towards developing any product or service. Moreover, due to the high costs associated with underwater tests, Forssea company is working towards building good simulators for their projects in all areas not just in autonomous docking.

Objective 4: Creating Technical Documentation on GitHub for Team Use

To ensure the reusability and accessibility of my work, it was one of the main objectives of my internship to create comprehensive technical documentation of my work. I would be responsible for documenting my research, simulation development process, POC implementation, and any other relevant information on dedicated GitHub repositories. By doing so, I would make my work available for the entire team to study, review, potentially contribute to, and reuse in future projects.

Objective 5: Designing a Procedure for Collecting Real-World Dataset

As I mentioned previously, there is a lack of information regarding the datasets and the behaviour of underwater dynamic docking garage. So a proposed solution is to collect datasets from the real garage through field testing. Such tests can be done on the physical robot and the docking garage in the company's workshop in the south of France in

Séte. As part of my internship, I will design a well-defined procedure for collecting the necessary dataset. This procedure will outline the steps involved in capturing data, such as sensor measurements, robot movements, environmental conditions, and any other relevant parameters. By establishing a systematic approach to data collection, I aim to enable the team to analyze and optimize the autonomous docking system by further analyzing the data collected from the garage.

1.3 Environment and Initial State

Forssea Robotics company has the main office in Paris and a workshop in Sète in the south of France. Since my internship is mainly about simulation and a proof of concept of algorithms to improve autonomous underwater docking, I was based in Paris. The working system at Forssea is hybrid so sometimes I can work remotely. To further improve and accelerate my performance, I used another PC at the company along with my laptop to distribute the computational costs. Moreover, the engineers at Forssea already developed a dynamic model of Argos ROV on the Gazebo simulator. Argos ROV is the ROV model developed by Forssea Robotics for light intervention missions. The gazebo model has the estimated dynamic properties and the imu, depth, and camera sensors. I would utilize this model and the newly developed model for the docking garage to simulate the autonomous docking scenario.

1.4 Contributions

1- Developing a simulation: One of the achievements in this work is developing a new simulation environment in Gazebo to simulate the disturbances acting on the garage. By simulating a realistic behaviour of the docking garage, we can later test any algorithms quickly as testing on real hardware is very expensive.

2- Developing a motion estimation algorithm: Another key achievement in this work was developing a motion estimation algorithm to estimate the garage's position despite loss in measurements. Having a robust and real-time estimated position of the garage would be very useful for implementing a docking strategy later.

3- Garage detection: To improve the perception and decision-making capabilities in the autonomous docking process, yolov5, a deep learning-based approach, was trained and implemented to detect the garage from different orientations and distances in real time. Having such a robust detection would be very useful in developing a control strategy to guide the robot towards the docking garage-.

4- Developing a control strategy for homing: Last but not least, a control strategy based on the garage detection algorithm was implemented. The goal of this algorithm was to guide the robot towards the garage and make sure the robot is facing the correct side that contains the entrance to the docking garage.

1.5 Outline of the thesis

In the first chapter of this work, a brief introduction to the work motivation and problem statement will be provided. Chapter 2 will talk about the planning and the resources. Then chapter 3 will summarize the relevant literature review while Chapter 4 will introduce the system analysis and design. Chapters 5 and 6 will talk about work development and results respectively. And last but not least, the conclusion and future work will be discussed in Chapter 7.

CHAPTER 2

Planning and resources evaluation

Contents

2.1 Planning 7
2.2 Resource Evaluation 7

2.1 Planning

This chapter will provide a quick overview of the planning and timeline followed in this work. Figure 2.1 demonstrates a Gantt chart summarizing the timeline during my internship. As we see in the figure, I started my internship by doing research about state-of-the-art parallel to developing the gazebo simulation. Then at a later stage, I started working on motion estimation. And by May, I started developing the garage detection model using deep learning which was followed by the control strategy for homing. In parallel to all these tasks, I was also working on documentation tasks.

2.2 Resource Evaluation

Regarding the available resources, the company's laptop is an i5 processor with 8GB of RAM and an NVIDIA Geforce GTX 960M with 2G of memory. To further improve and accelerate the performance, especially while running the simulation, I connected my laptop to another PC at the company. This PC has an i7 processor with 16GB of RAM and 4GB of GPU. I used to connect the two machines and run the simulation on the PC and the scripts on the laptop.

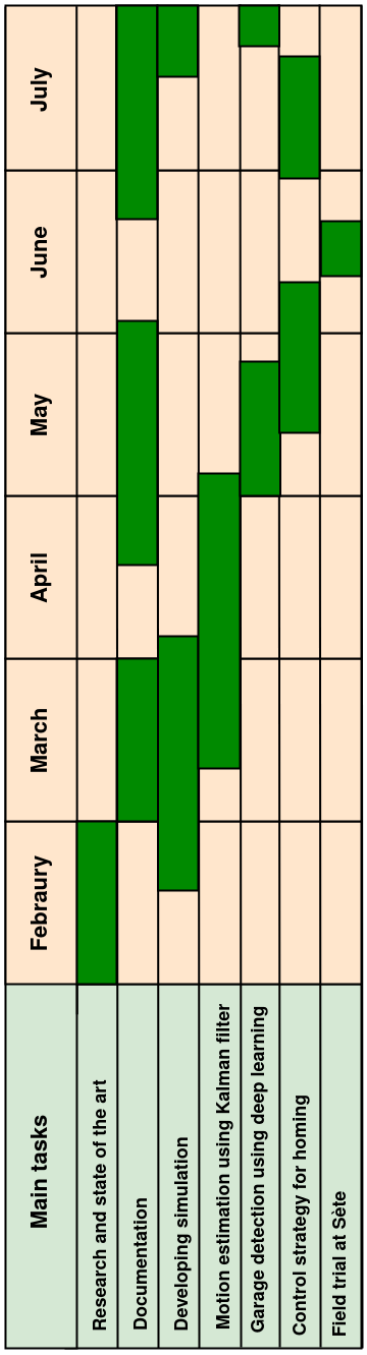


Figure 2.1: Example of a Gantt chart (made with Kplato)

CHAPTER 3

Literature review

Without history we are nothing, so it is worth finding out something about it - Keith Allen

Contents

3.1 Underwater autonomous docking **9**
3.2 Inspirations from other fields **13**
3.3 Control strategies for underwater autonomous docking **15**

3.1 Underwater autonomous docking

Nowadays, there are a lot of developments in the area of underwater robotics to satisfy the growing need. There are two main types of underwater robotics: Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs). The basic idea of ROVs is that they are controlled by human pilots from the surface and are connected through a tether that provides power and communication. The ROVs are often equipped with cameras, lights, and a wide range of specialized sensors and tools which allow them to perform challenging operations such as underwater inspections, maintenance, repair of offshore oil rigs, cable laying, collecting samples, and even archaeological excavations. The main users of ROVs are in industries like oil and gas, underwater inspection, and underwater construction. The growing utilization of ROVs as well as their reliability have allowed more cost-effective and efficient operations in hazardous underwater environments. On the other hand, one of the main types of underwater robots is the Autonomous Underwater Vehicles (AUVs) which are designed to operate autonomously without real-time human control. These vehicles are often equipped with various sen-

sors such as Inertial Navigation System (INS), Doppler Velocity Log (DVL), cameras, etc. They often have powerful computing units to allow the implementation of artificial intelligence algorithms that enable them to navigate autonomously, collect data, and make decisions based on the surrounding environment[16]. One of the main challenges associated with using ROVs and AUVs is retrieving them after the mission. This process can be done through docking. There are three main types of docking which depend on the docking garage/station. Each docking type has its own pros and cons.

3.1.1 Static docking

The first docking type and the most common one is static docking where the docking station is stationary. Despite the complex task of performing docking in general, this is one of the easiest docking scenarios because a lot of disturbances are ignored. Figure 3.1 illustrates an example of static docking [23].

Figure 3.1: A demonstration of static docking [23]

3.1.2 Towed docking

Another type of docking is towed docking where the docking station is being towed by a surface vessel at a constant speed. By moving at a constant speed, the towed docking station may be considered stationary where external disturbances can be ignored. Figure 3.2 demonstrates an example of towed docking [23].

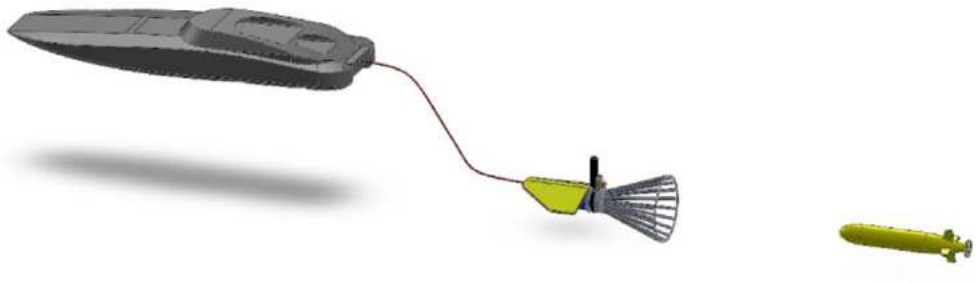


Figure 3.2: A demonstration of towed docking [23]

3.1.3 Dynamic docking

This is the most challenging type of docking because the docking station is subject to many external disturbances. The docking station is literally hung by a cable and if there is an efficient heave compensation system on the surface vessel, the docking station will be subject to heave oscillations which is a major challenge even for experienced pilots. Figure 3.3 explains how the vessel's movements can be transmitted to the docking station. Such movements are highly affected by the sea state. Moreover, figure 3.4 illustrates the main disturbances acting on the docking stations which are primarily the heave and yaw disturbances and in some cases the pulling force by the tether itself.

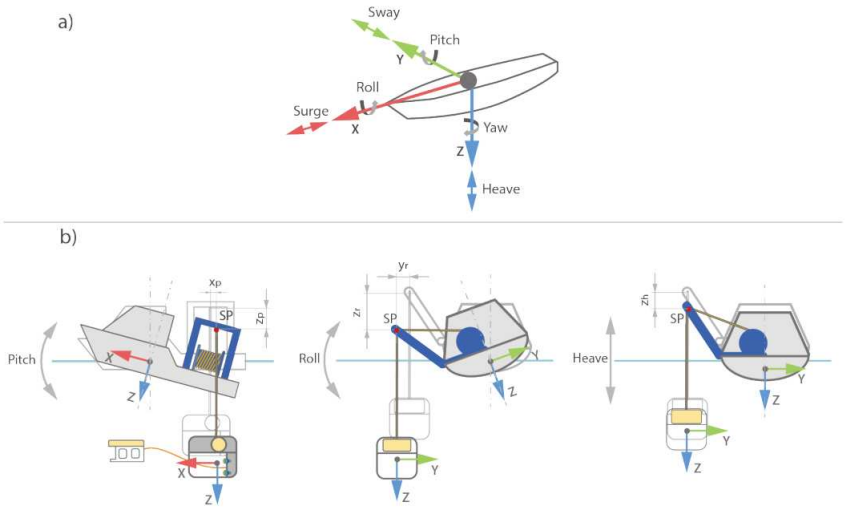


Figure 3.3: How the vessel's movements are transmitted to the docking garage [21]

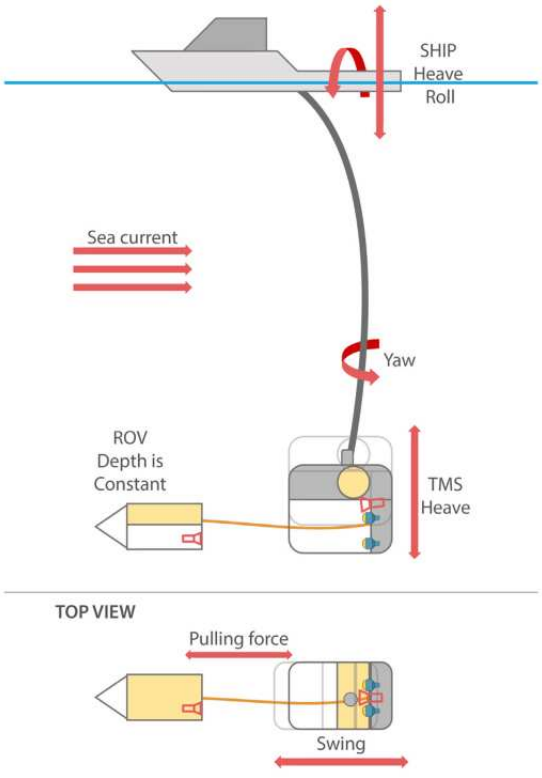


Figure 3.4: The disturbances affecting the vessel and the garage [19]

3.2 Inspirations from other fields

Since there was not much in the literature review regarding underwater non-stationary autonomous docking, a good approach was to read through similar problems in other fields to get ideas and inspiration. One of these inspirations was found in the work of Wang et al. on aerial recovery of unmanned autonomous vehicles (UAVs) [22]. The authors proposed a novel approach for visual navigation to perform aerial recovery of UAVs by safely guiding the UAV toward the docking probe until the docking is completed successfully. Figure 3.5 illustrates the aerial docking scenario where the docking drogue is towed by the aircraft and, of course, it will be subject to external disturbances. This scenario is very similar to underwater non-stationary docking except that the latter includes significant heave oscillations which adds more complexity to the docking operation. The solution proposed in this work consists of detecting the docking drogue using deep learning and then applying an adaptive region of interest (AROI) tracker to rapidly determine the region of interest (ROI) which in this case will be the drogues itself. Afterwards, the proposed framework would obtain the centroids of the markers and then using stereo vision, the drogue's pose can be calculated. Moreover, the framework also includes a Kalman filter to achieve a more accurate estimation of the drogue's pose. Figure 3.6 shows an example of drogue detection using deep learning. Figure 3.7 shows the proposed yolov3 architecture that was implemented to detect the drogue's pose in real-time.

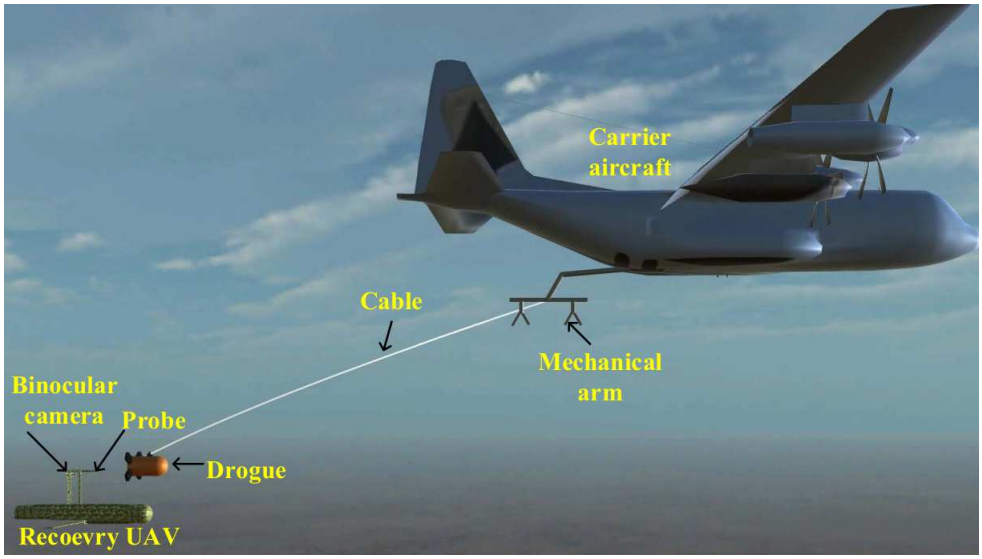


Figure 3.5: Aerial docking scenario

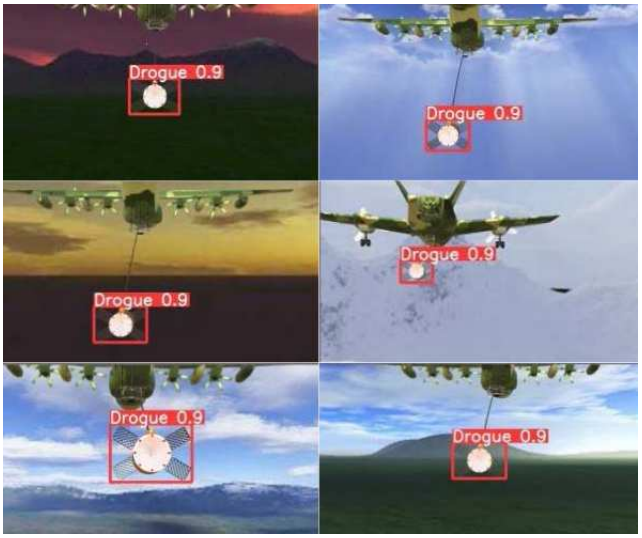


Figure 3.6: Drogue detection using deep learning

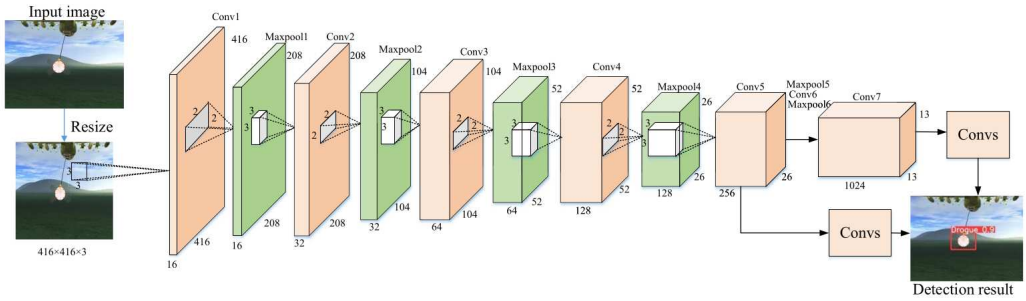


Figure 3.7: The proposed yolov3 deep learning model to detect the drogue

3.3 Control strategies for underwater autonomous docking

One of the recent works on underwater autonomous dynamic docking was conducted by Trslic et al. from the University of Limerick [19]. Their work included a vision-based solution for underwater autonomous docking which, up to the author's knowledge at that time, was the first work to tackle this particular problem. In the article, they proposed a machine vision-based docking system to utilize subsea camera pose estimation and a known light marker pattern for standard work-class ROVs with suspended cage-type Tether Management System (TMS). The system's performance was validated through real-world tests conducted in the North Atlantic Ocean, demonstrating comparable results with a commercial state-of-the-art underwater navigation system. Figure 3.8 illustrates the image processing steps needed before obtaining the centroids of the light beacon installed on the TMS. Figure 3.9 summarizes a flowchart for the safety checks to perform autonomous underwater docking. The core idea of these checks is to make sure that the estimated pose value makes sense and did not diverge. To do that, they count the number of detected beacons and then check the value difference between the newly calculated pose and the previous pose to make sure there has been no divergence. The author in this work also highlighted an important point which is that because of the high inertia, big ROVs with larger mass respond slowly to thruster commands, making it impractical to fully compensate for the docking station's (DS) heave motion. As a result, dynamic docking unavoidably involves contact between the vehicle and the DS, in contrast to static docking, where minimal contact occurs. The DS heave motion during the experiment is depicted in Figure 3.10.

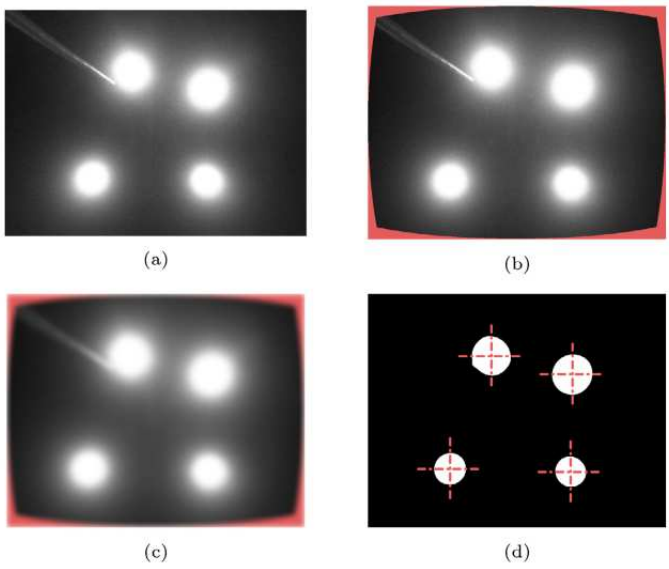


Figure 3.8: Image processing steps to obtain the centroids of the light beacons

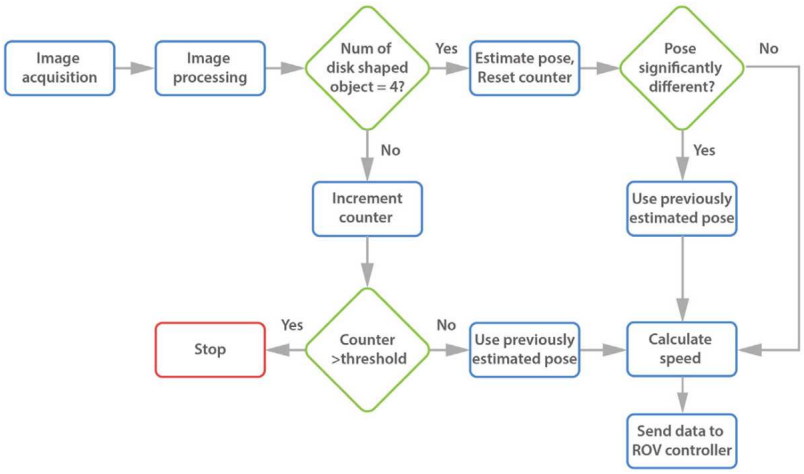


Figure 3.9: A flowchart of safety checks to perform autonomous underwater docking

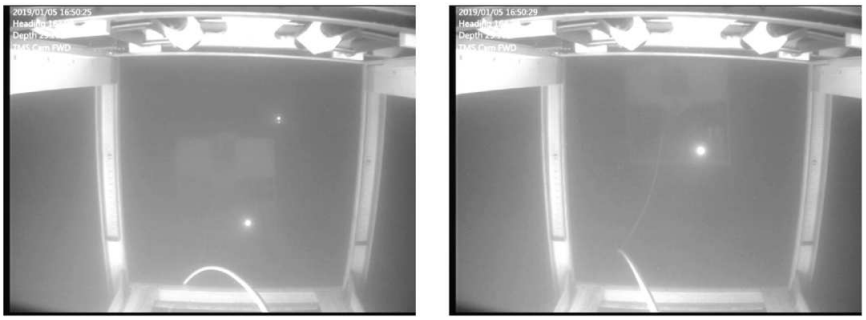


Figure 3.10: The DS heaving while the ROV holds constant depth [13]

Another article by Trslić et al from the University of Limerick proposed a novel Neuro-Fuzzy approach for dynamic position prediction as another step towards underwater dynamic autonomous docking [21]. They proposed a method for predicting the heave motion of the docking station. The proposed method is based on the Adaptive Neuro-Fuzzy Inference System (ANFIS). In general, Work-class ROVs face challenges such as limited onboard power, high hydrodynamic drag forces, and inertia which make it difficult for them to match the heave motion of a docking station that is suspended from a surface vessel and is subject to all kinds of disturbances. As a result, current docking procedures heavily rely on the experience of ROV pilots to estimate the heave motion and decide when to dock. Inspired by how the human pilots would do, the author proposed an ANFIS network to predict the future docking station's position and then later determine when to dock. They collected a real dataset from a previous trial in North Atlantic Ocean in January 2019. They attached a depth sensor to the docking

station to collect the dataset for training. Figure 3.11 illustrates the dataset utilized for evaluating the ANFIS wherein part (a), the initial 200 seconds of data was used for training while the 50 seconds were used for validation and checking. In part (b), the performance of the ANFIS model is evaluated by predicting the TMS depth between 1 second and 3 seconds into the future. They found that the maximum period of prediction into the future while maintaining good accuracy was 2.5 seconds. Figure 3.12 illustrates the heave motion of the TMS the blue shaded area indicates the ideal docking position which corresponds to the minimum TMS heave speed as it is about to change its speed direction during the vertical oscillations. So by predicting the future TMS position, we can plan when to dock given the low dynamics of the ROV and the high oscillations of the TMS.

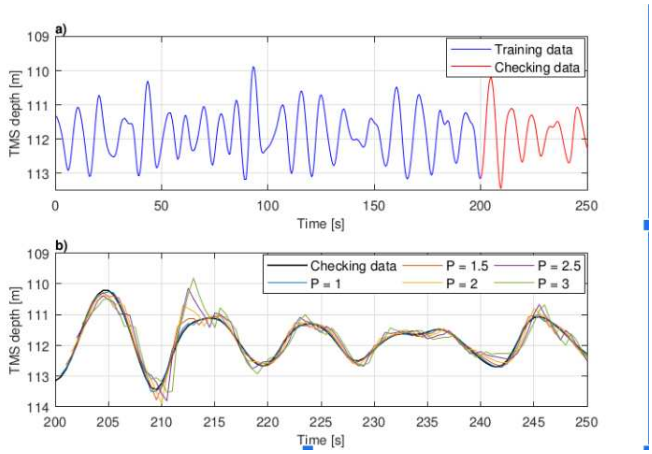


Figure 3.11: The dataset used for training the ANFIS as well as its prediction performance

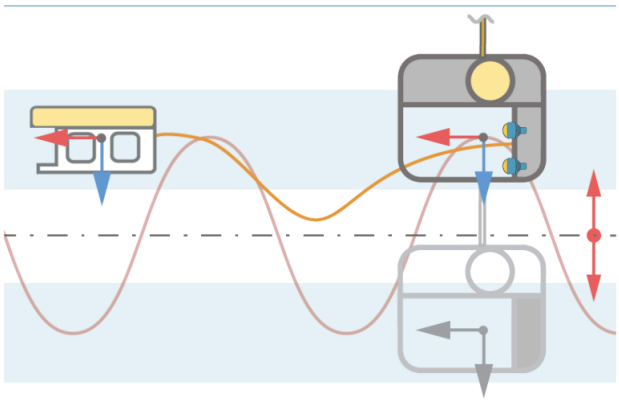


Figure 3.12: The heave motion of the TMS and when it is the best time to dock

Chapter 4

System Analysis and Design

If the big rocks don't go in first, they aren't going to fit in later - Stephen R. Covey

Contents

4.1 Docking scenario and challenges **19**
4.2 System overview **22**
4.3 Software and tools used **24**

This chapter provides an overview of the docking scenario and the associated challenges, an overview of the proposed solution as well as a summary of the software and tools used.

4.1 Docking scenario and challenges

This section will provide an overall summary of the specific case study of docking discussed in this report as well as the challenges and limitations. First of all, the docking scenario is non-stationary docking which means that the docking garage will be affected by disturbances. There are, of course, some assumptions that have been made at the beginning of this work such as:

1- Vertical oscillations: Based on the available information in the literature review as well as the Forssea team experience, it was assumed that there would be vertical disturbances acting on the docking garage with amplitudes ranging from 0.25m up to 2m depending on the sea state. Of course, such disturbances can be decreased by using

a good tether management system (TMS) to allow for heave compensation but this is not the case most of the time.

2- Heading oscillations: Based on the available information in the literature review as well as the Forssea team experience, it was also assumed that the docking garage will change its orientation due to the cable tension but the frequency and amplitudes of such movements are assumed to be low but it still needs to be taken as it can prevent the docking if the robot could not face the side of the markers that contain the entry to the garage.

3- Pitch and roll oscillations: It was assumed that pitch and roll oscillations of the garage can be neglected. Collecting real datasets of the garage will support this assumption which will be discussed in later chapters.

4- Argos ROV has an accurate INS system: It was also assumed that Argos ROV has an accurate inertial navigation system (INS) that will enable it to keep track of its pose and be able to return to the starting point which will be the docking garage. Of course, due to drift errors, and environmental disturbances, the garage may not be in the exact position when the ROV left it. But at least having an accurate INS on board the Argos ROV can guide it to the vicinity of the docking garage. Fortunately, this assumption was considered true because the Argos ROV has Rovins INS installed onboard which is a high-quality fibre-optic gyroscope (FOG)-based INS for subsea vehicles.

Figure 4.1: Rovins INS sensor manufactured by iXblue [8]

5- Docking garage has markers: It was also assumed that the docking garage has aruco markers to allow for visual localization of the docking garage with respect to the Argos ROV when the markers are detected. Figure 4.2 illustrates the attached markers in reality and in simulation. As we can see, there two main sets of markers: one set on the outer entrance to allow the robot to localise the garage and another set of markers inside the garage to allow the robot to do fine tuning and smooth localisation.

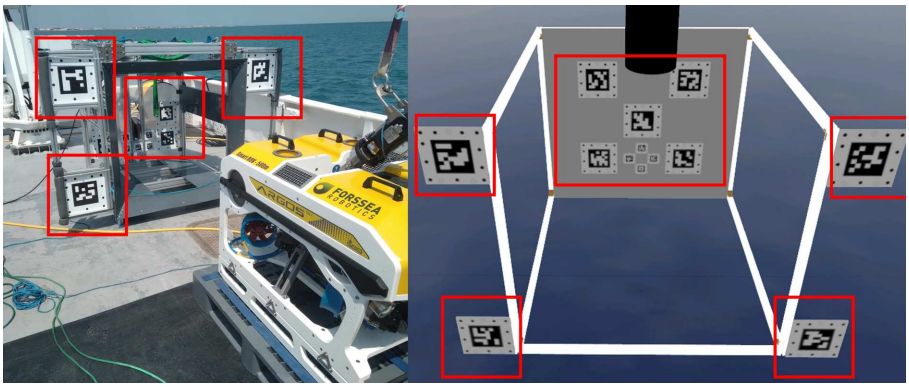


Figure 4.2: Visual markers added to the docking garage

challenges

1- heavy ROV dynamics

One of the major challenges in dynamic underwater docking is that the ROV has slow dynamics compared to oscillating docking garage. This results in the incapability of the ROV to match the garage's movement and thus the need for more intelligent behaviour by predicting the future position of the garage and planning accordingly.

2- bad detection due to bad visibility

Figure 4.3 shows an example of perfect docking conditions where the visibility is great and the docking garage wasn't oscillating much.

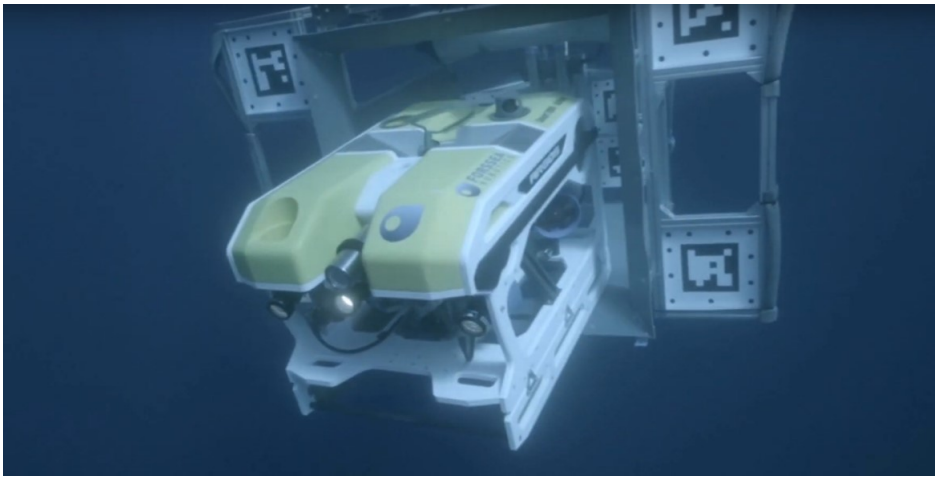


Figure 4.3: An example of docking in very good conditions from a previous trial in La Ciotat, 2020 [1]

4.2 System overview

This section will provide an overall summary of the main components of the proposed solution. There are two main phases in the docking process: Homing and Docking. Homing is where the vehicle tries to get close to the docking station. Docking is when the vehicle is already in the vicinity of the docking station and starts getting closer to the final docking. The proposed solution contains the following steps:

1- Go to the last know location of the garage: In this step the Argos ROV will go to the last know location of the garage before it has left it. This step depends on the INS installed on the Argos ROV.

2- Search for the garage: After reaching the last known location of the garage, the robot will keep looking for the garage by rotating itself and the camera while trying to detect the garage using a deep learning model (yolov5). In an ideal world, the docking garage will still exist in the last known location but due to drifts in the INS and disturbances acting on the docking garage, this is mostly not going to happen.

3- Approach the garage: Once the Argos ROV can detect the garage using deep learning, it will start approaching it. It will do that by tracking the detected garage and making it in the center of the image while going forward.

4- Face the markers' side of the garage: Once the robot is close enough to the garage, it will start rotating around it with the aim of facing the markers' side of the garage which is the correct side for entering the garage and conducting the docking. So basically, the robot will keep rotating around the garage until it detects the markers and calculate the garage's pose and know its relative orientation.

5- Observe the garage's pose: After approaching the garage and facing the side of the markers, the ROV will observe the garage movements and record its pose thanks to the detected markers. The implemented motion estimation algorithm will use the recorded garage's behaviour to optimize its parameter and be able to estimate the garage pose when there is no detection.

6- Predict the garage's pose: The next step after recording the garage's behaviour and tuning the motion estimation parameters is to estimate the garage pose in future time steps.

7- Decide when to dock: Based on the future garage pose estimation, the ROV will decide when to dock. The best time to dock is when the docking garage is about to change its motion direction while oscillating up and down because at that time step the vertical speed of the garage is zero.

8- Start docking: The final step would be to start docking based on the output of the motion estimation algorithm.

Figure 4.4 summarizes the main steps in the proposed solution for conducting autonomous docking for a non-stationary floating docking garage.

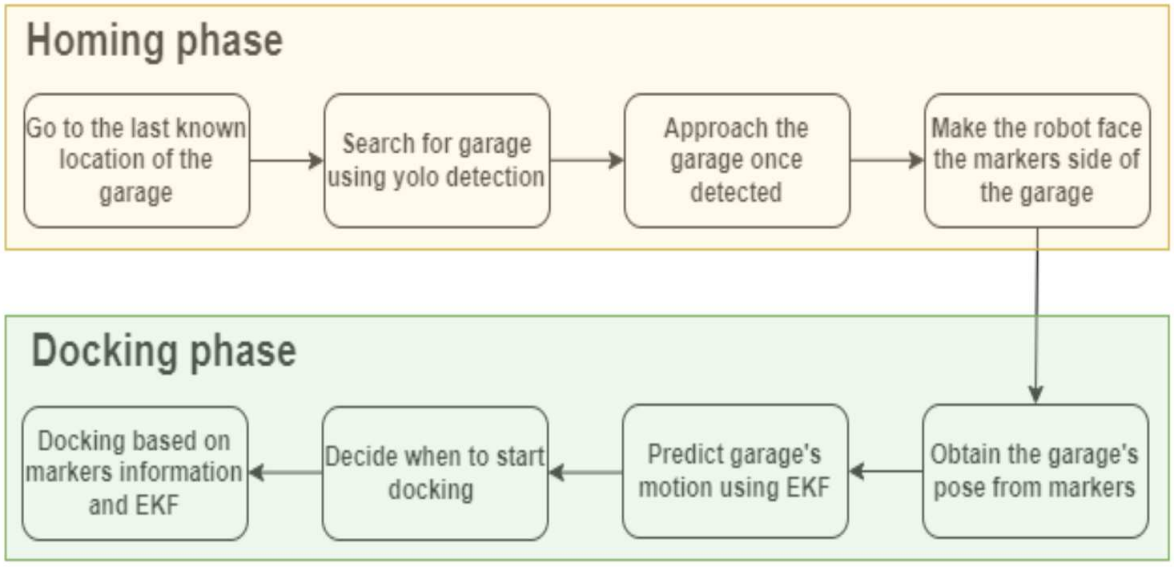


Figure 4.4: Overall summary of the complete proposed solution

4.3 Software and tools used

This section will provide a summary of the main software and tools used in this work as well as the overall layout and the connections between them.

4.3.1 Robot Operating System (ROS2)

ROS2 is a powerful framework for developing and controlling robots and robotic systems. It has more features than its predecessor ROS. ROS2 is designed to be more robust, scalable, and efficient. One of the main advantages of ROS2 is that no central node is required like in ROS which opens the door for more multi-robot collaboration [7]. ROS2 was used heavily within the scope of this work as the middleware connecting all the different nodes together.

4.3.2 Gazebo simulation

Building the gazebo simulation for the docking scenario is the cornerstone of this project because testing any of the proposed algorithms directly on the real system will be expensive and not efficient. Gazebo sim is based on the software conception of entity-component system architecture "ECS" where entity refers to any element or "object" within the simulated world, such as models, links, collisions, visuals, lights, joints, and more. Each entity is identified by a numeric ID and can have multiple components associated with it. The entity IDs are assigned dynamically during the runtime of the simulation. A component is responsible for adding specific functionality or attributes, such as pose, name, material, and more, to an entity. Ignition Gazebo provides a range of pre-existing components, including Pose and Inertial, which can be readily utilized. Additionally, developers can create their own components either by inheriting from the BaseComponent class or by instantiating a Component template. Gazebo works through two main parts: back-end server and front-end client. You can run the back-end server without the front-end client but in this case, there will not be a GUI to visualize what is happening but gazebo topics are being published in the background as usual. The idea of having a separate client and server can be useful in having distributed computations. Figure 4.5 summarizes the working architecture of Gazebo simulation [2]

4.3.3 Command Line Interface (CLI)

CLI is a text-based user interface (UI) that can be used to run programs and interact with the computer. Within the scope of this work, CLI was used to run the gazebo simulation and ros2 nodes. It was also used to debug and monitor ongoing operations.

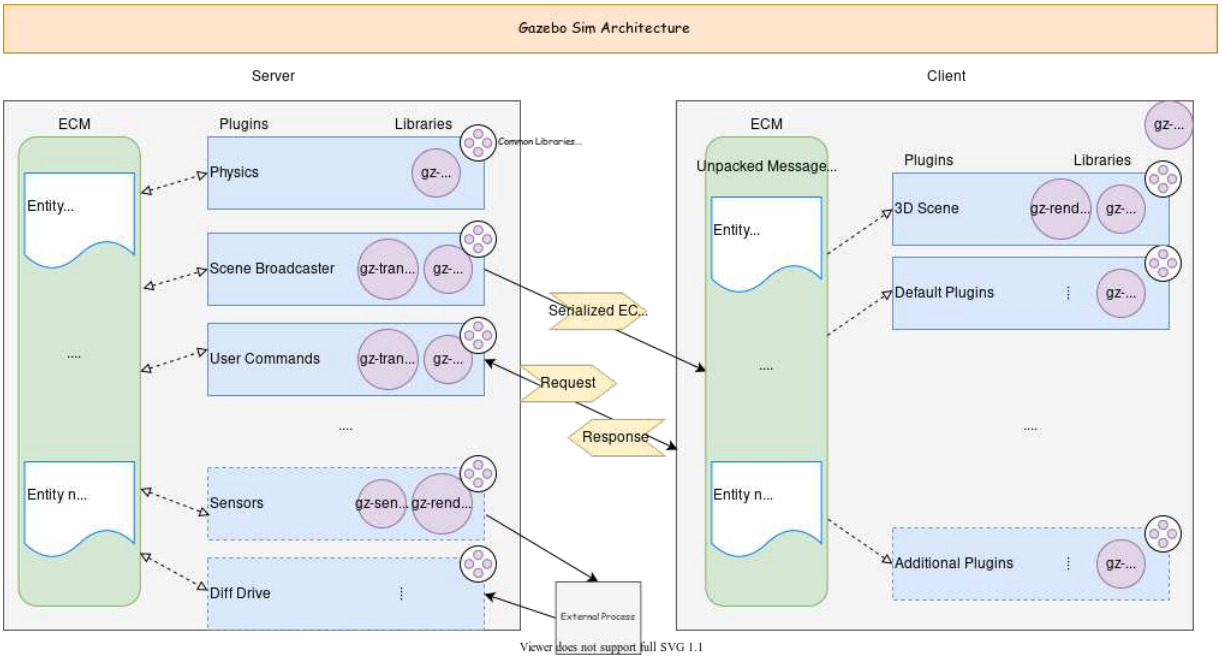


Figure 4.5: Gazebo sim architecture [2]

4.3.4 Plotjuggler

Plotjuggler is a very efficient tool to visualize time series data in both online and offline modes. It can work with ROS and ROS2 by subscribing to the topics. You can also upload rosbags for offline analysis. Plotjuggler can even be used to re-publish messages and visualize them in RViz. In this work, plotjuggler was used in analysing the output of the motion estimation algorithm as well as the garage movement. In addition to simply plotting the data from ros topics, plotjuggler can also perform simple calculations and filter such as derivatives, moving average, outlier removal, etc. Figure 4.6 shows the interface of plotjuggler and the different options available.

4.3.5 rqt

rqt is a Qt-based framework for GUI development for ROS. It can be used to monitor, analyse, and debug the system. It has several functionalities such as publishing ros topic, image topic visualization, plotting and most importantly the node graph that shows all the nodes and the connections between them. With such features in one place, rqt is an essential tool for any ros developer. Within the scope of this work, rqt was used to visualize the camera topic as well as visualise the node graph to debug any in the topics or nodes. Figure 4.7 shows the interface of rqt as well as the different options available.

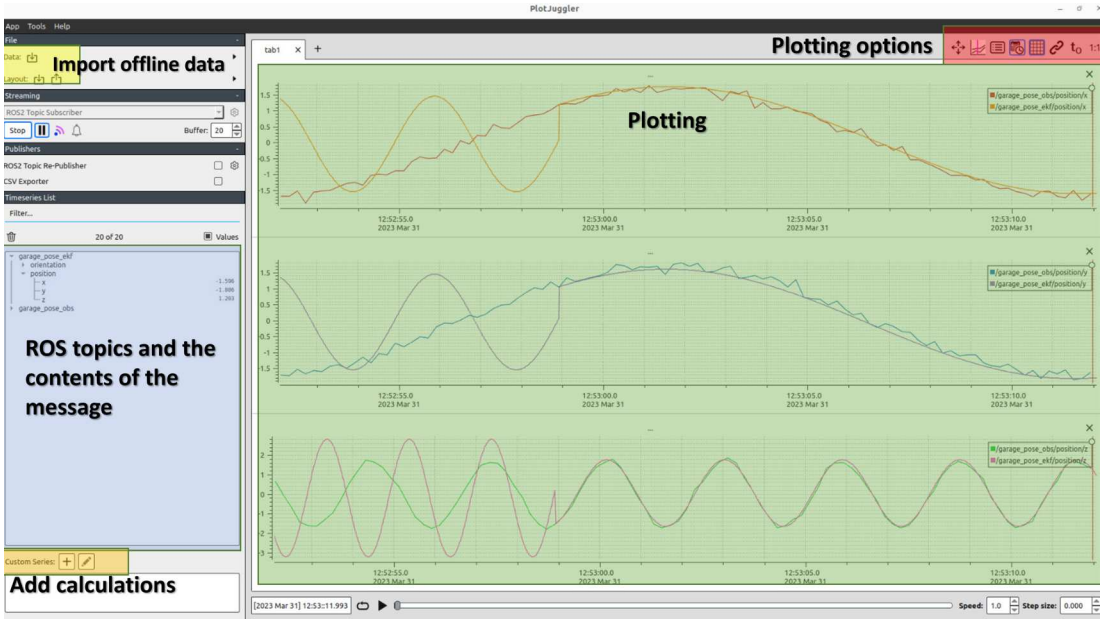


Figure 4.6: An overview of plotjuggler interface

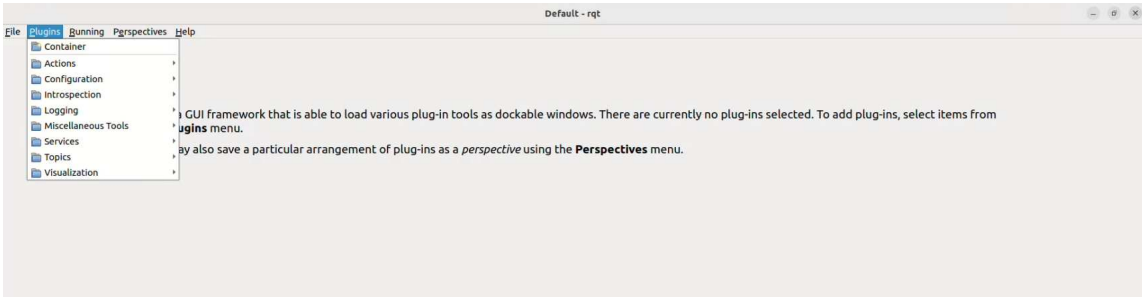


Figure 4.7: An overview of rqt interface

To sum up, in order to implement and test the proposed solution in this work, several software and tools were used as explained in the previous sections. Figure 4.8 summarizes the software and tools used and their connections.

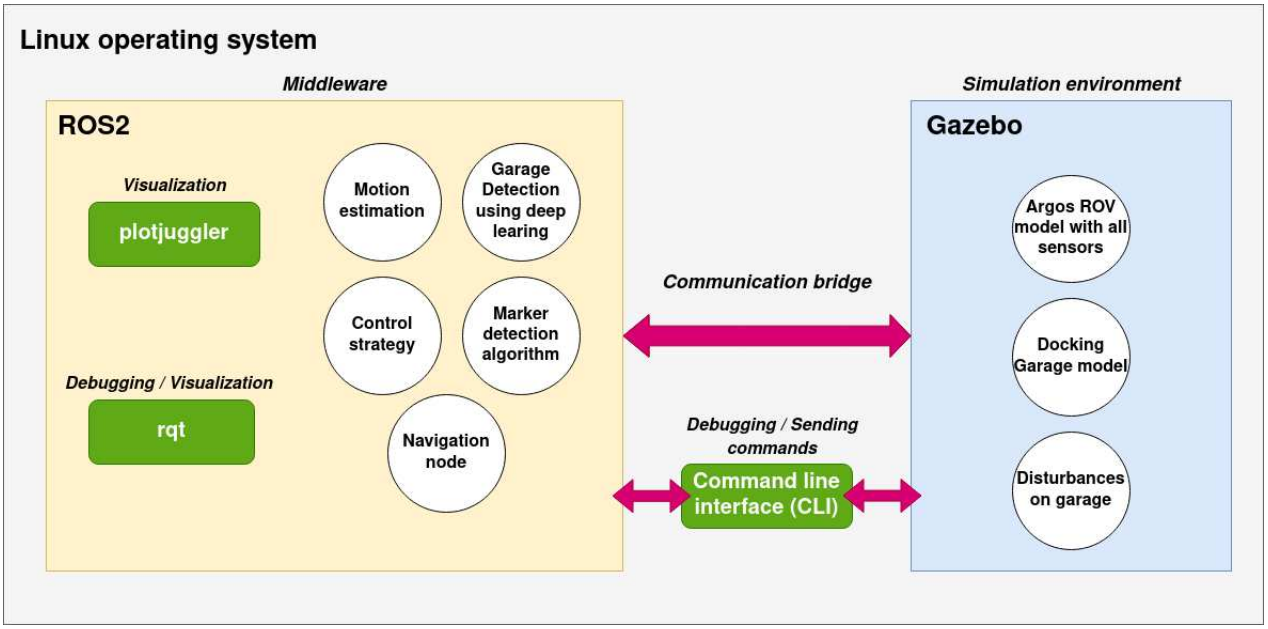


Figure 4.8: Overall software and tools used in this work

Chapter 5

Work Development

All models are wrong but some of them are useful - George Box

Contents

5.1 Building the simulation **29**
5.2 Homing - Garage detection **36**
5.3 Homing - Control strategy **39**
5.4 Docking - Motion estimation **41**
5.5 Collecting datasets from the real garage **48**

This chapter will discuss the work development as it will focus on the main work directions and elements that have been implemented to improve autonomous docking for ROVs. This chapter is important because it provides an overview of the main milestones as well as their importance within the scope of autonomous docking for ROVs. This chapter will discuss the main milestones implemented in this work. As explained before in section 1.2, the main objective of this work is to improve the perception and control strategies in autonomous docking operations for ROVs. Figure 5.1 summarizes the main work directions in this work that all fall under the big umbrella of improving the perception and control of autonomous docking for ROVs.

5.1 Building the simulation

Developing a good simulation for the docking scenario is important and very helpful in developing and testing docking algorithms. It is very expensive in terms of money and time to test the algorithm directly on the robot. On the hand, there are simulation environments for marine robotics but none of them simulated the scenario of a floating

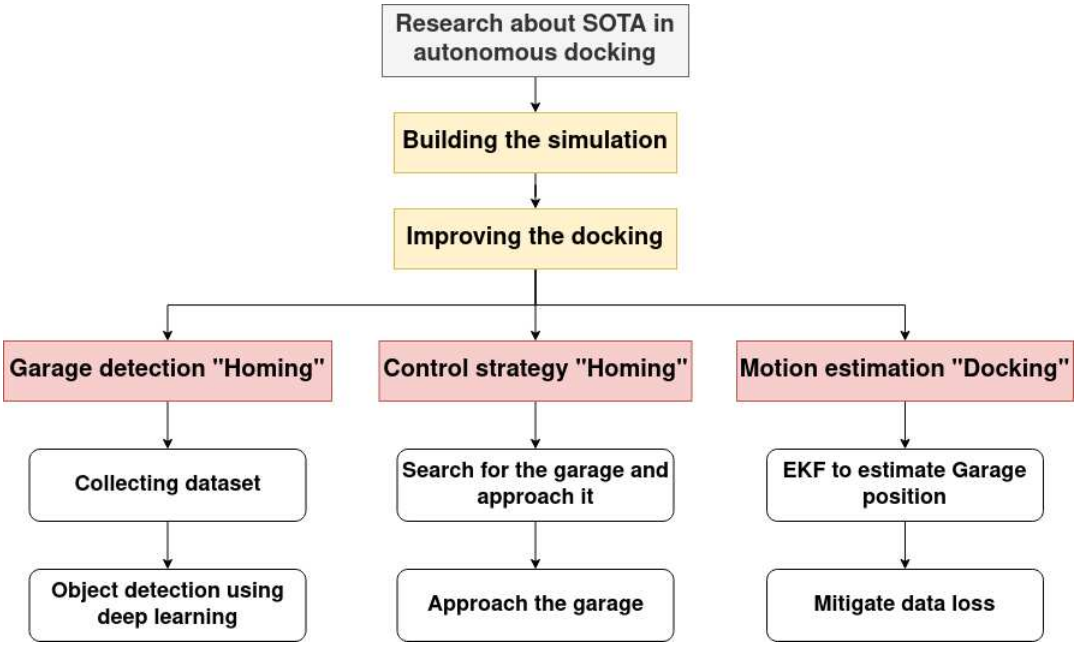


Figure 5.1: Summary of the work directions discussed in this work

docking garage The basic idea of building a simulation for the docking scenario is to simulate the waves, currents, and cable disturbances since the focus of this work is on floating docking garage which is highly subject to these disturbances. To build the simulation environment for the docking scenario in the gazebo sim, there were two main approaches: first, a cable model would be implemented and attached to the garage and then add disturbances assuming the garage is hung via a cable. The second approach is to apply all the forces and torques without any cable attached to the garage. After consideration, the first approach was chosen because the second approach is complicated in terms of calculations. In the first approach, the garage is already hung via a cable which is similar to the real scenario and then later, disturbances could be added. In gazebo sim, we utilize some of the existing plugins for hydrodynamic modeling such as added mass, drag, and buoyancy [14]. So by using these plugins, there would not be a need to model everything from scratch. Figure 5.2 illustrate the first approach of hanging the garage via a cable. As illustrated in the figure, the cable is modelled using two rigid links that are connected to each other via a prismatic joint with some damping. The cable is connected to the garage and the surface vessel through ball joints to allow rotations in all directions.

Moreover, when it comes to adding the disturbances in the garage, three approaches have been implemented as explained in the following paragraphs.

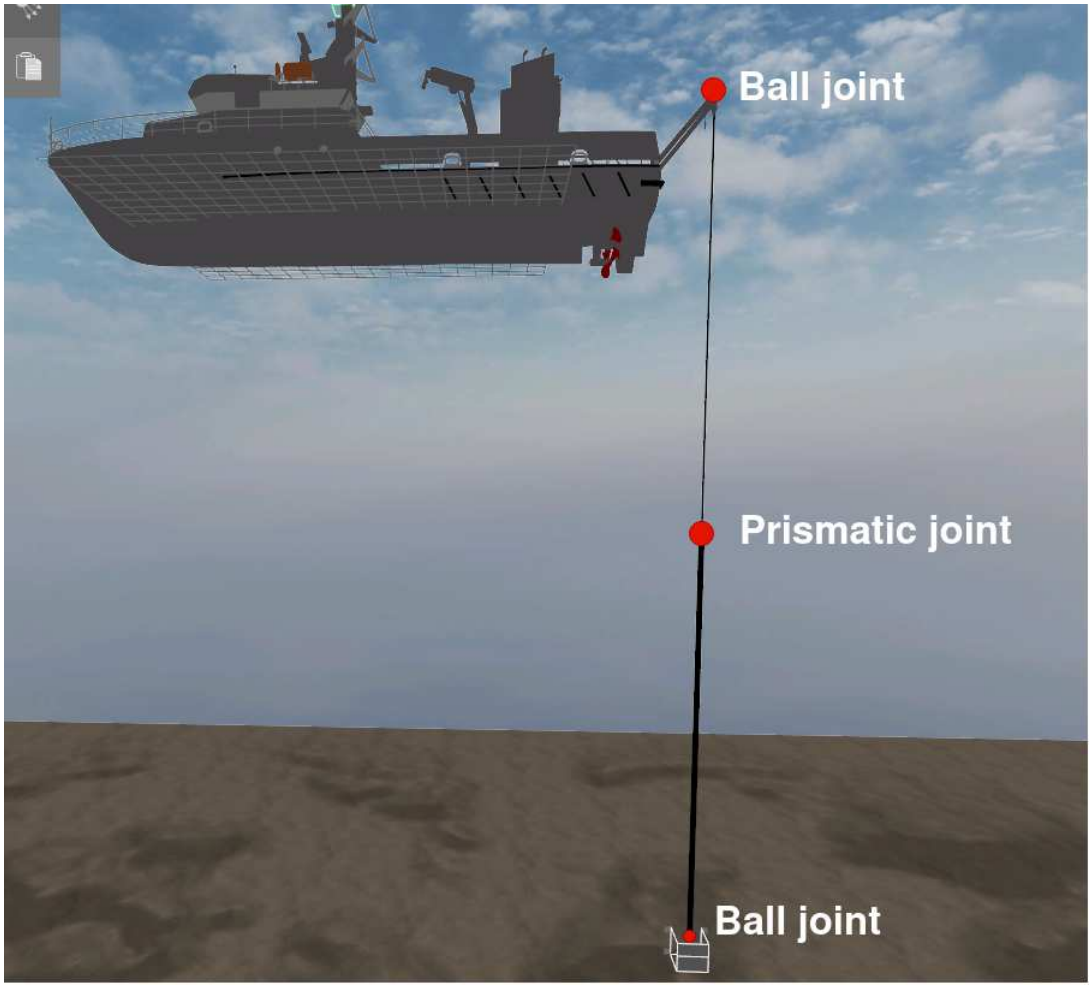


Figure 5.2: An illustration of the connections between the cable and the docking garage

5.1.1 Integrating wave sim package

One of the available packages to simulate waves and surface vehicles is asv wave sim[17]. The main motivation for using this package is to simulate realistic wave behaviour and let the resulting forces and torques acting on the surface vessel transmit to the surface vessel and then to the docking garage via the cable. So in the end, we would have a realistic behaviour of the docking garage. Figure 5.3 shows a screenshot of the final wave mesh in the Gazebo simulation. This is what will happen in reality. Wave sim package is an open-source package that contains three main plugins:
- gz-waves1-waves-model-system: This plugin is responsible for simulating the waves model and calculating the resulting forces and torques. It has several parameters to tweak such as (tile size) to specify the size of the whole wave mesh. It also contains (cell count) to specify the number of cells or in other words the resolution. Of course, the

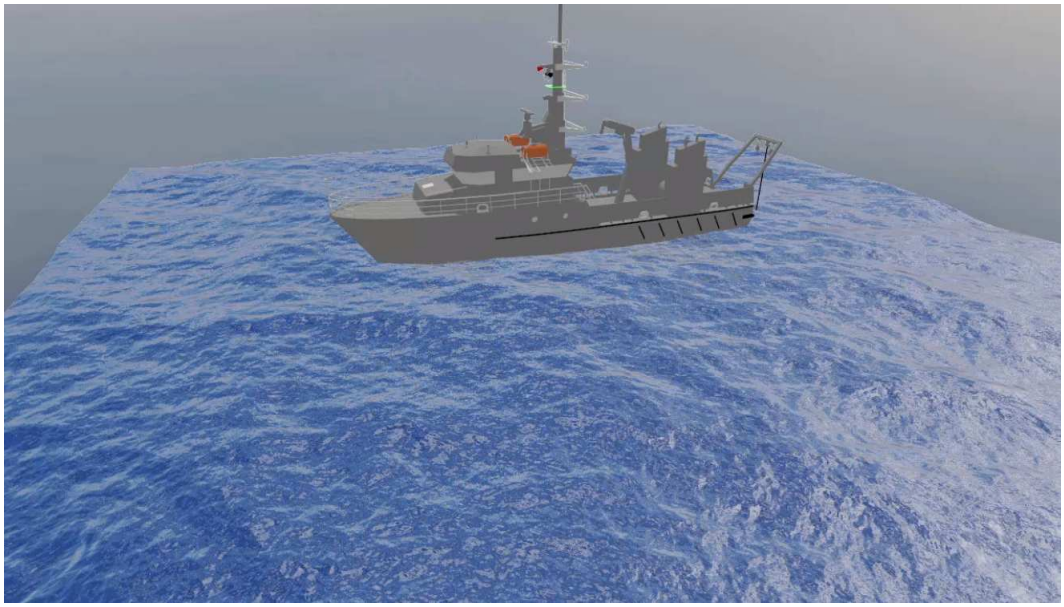


Figure 5.3: Implementation of wave sim plugin in Gazebo sim

higher the number the more accurate calculations but the more expensive in terms of computation costs. The plugin also has parameters such as (algorithm), (wind speed), (wind angle deg) to specify the algorithm to be used, the wind speed, and the wind direction respectively. The algorithm types to be used are sinusoid, trochoid and fft. Figure 5.4 illustrates some of the parameters to tweak in the gz-waves1-waves-model-system plugin.

```
<tile_size>256.0 64.0</tile_size>
<cell_count>128 32</cell_count>

<!-- Wave algorithms
  - These elements specify the wave generation method
    and wave spectrum parameters.
-->

<!-- Either: `fft` waves parameters -->
<algorithm>fft</algorithm>
<wind_speed>5.0</wind_speed>
<wind_angle_deg>135</wind_angle_deg>
<steepness>2</steepness>
```

Figure 5.4: Some of the parameters in gz-waves1-waves-model-system plugin

- gz-waves1-waves-visual-system: This plugin is responsible for generating the visual wave mesh. So it is just visualization; it can be disabled and there will be no wave

visual but the effect of the waves will appear thanks to the gz-waves1-waves-model-system plugin. gz-waves1-waves-visual-system contains almost the same parameters as in gz-waves1-waves-model-system plugin. It has an additional parameter called (the mesh deformation method) to control the shading algorithm of the wave visual; there are two algorithms options: dynamic geometry and dynamic texture.
- gz-waves1-hydrodynamics-system: This plugin is responsible for calculating the forces and torques based on the hydrodynamic parameters so it should be attached to the surface object that you want to simulate; in our case, it is the surface vessel. This plugin is very similar to the hydrodynamics plugin developed by gazebo [5]. The difference is that gz-waves1-hydrodynamics-system is more designed for surface vehicles, unlike gazebo hydrodynamics which is designed for underwater vehicles. Figures 5.5 and 5.6 illustrate some of the parameters needed in both plugins.

```
<!-- Hydrodynamics -->
<hydrodynamics>
  <damping_on>1</damping_on>
  <viscous_drag_on>1</viscous_drag_on>
  <pressure_drag_on>1</pressure_drag_on>

  <!-- Linear and Angular Damping -->
  <cDampL1>1.0E-6</cDampL1>
  <cDampL2>1.0E-6</cDampL2>
  <cDampR1>1.0E-6</cDampR1>
  <cDampR2>1.0E-6</cDampR2>

  <!-- 'Pressure' Drag -->
  <cPDrag1>1.0E+2</cPDrag1>
  <cPDrag2>1.0E+2</cPDrag2>
  <fPDrag>0.4</fPDrag>
  <cSDrag1>1.0E+2</cSDrag1>
  <cSDrag2>1.0E+2</cSDrag2>
  <fSDrag>0.4</fSDrag>
  <vRDrag>1.0</vRDrag>
</hydrodynamics>
```

Figure 5.5: The parameters needed in gz-waves1-hydrodynamics-system plugin [17]

Despite having a very nice visualization, there are some limitations such as the need to accurately tweak the parameters to have a realistic simulation. Moreover, there was a plugin conflict when this package was implemented. Later, when the Argos ROV model was added to the simulation, the buoyancy plugin was not working properly. The reason is that the Argos ROV model utilizes the hydrodynamic plugin developed by gazebo [5] while the surface vessel model utilizes the hydrodynamic plugin from wave sim [17]. This conflict resulted in the inability of the thrusters to work and move the robot as expected. So perhaps a future improvement is to solve this issue and be able to use both plugins in the same simulation.

```
<plugin
filename="ignition-gazebo-hydrodynamics-system"
name="ignition::gazebo::systems::Hydrodynamics">
    <link_name>base_link</link_name>
    <xDotU>-4.876161</xDotU>
    <yDotV>-126.324739</yDotV>
    <zDotW>-126.324739</zDotW>
    <kDotP>0</kDotP>
    <mDotQ>-33.46</mDotQ>
    <nDotR>-33.46</nDotR>
    <xUU>-6.2282</xUU>
    <xU>0</xU>
    <yVV>-601.27</yVV>
    <yV>0</yV>
    <zWW>-601.27</zWW>
    <zW>0</zW>
    <kPP>-0.1916</kPP>
    <kP>0</kP>
    <mQQ>-632.698957</mQQ>
    <mQ>0</mQ>
    <nRR>-632.698957</nRR>
    <nR>0</nR>
</plugin>
```

Figure 5.6: The parameters needed in gazebo hydrodynamics plugin [5]

5.1.2 Utilizing the buoyancy plugin

Another approach that was implemented to simulate the disturbances on the docking garage is to utilize a bug in the buoyancy plugin in the gazebo simulator. The buoyancy plugin simply works by adding the buoyancy forces on all objects in the simulation that have a volume following Archimedes's principle [4] [11]. Theoretically speaking, if you have a simulation with underwater/surface vehicles with the buoyancy plugin without any external forces, the object will float/sink and be in a stable state after some time. However, in our case, the surface vessel was never stable even though the center of gravity was below the center of buoyancy. A potential reason for this might be because the mesh model of the surface vessel was considerably large; the order of size was in tens of meters. Another reason might be because the volume mesh of the surface vessel in the simulation was a simple 3D rectangle for simplicity. According to [10], the floating object with a semicircular hull shape is more stable than the ones with rectangular/ square shapes because, in the latter, the center of buoyancy can change considerably. Figures 5.7 and 5.8 illustrate the difference between the two configurations; in our implementation, the volume mesh was rectangular such as Figure 5.7. Anyway, the vessel was never able to stabilise leading it to always oscillate. These oscillations are transmitted to the docking garage via the cable which simulates realistic disturbances on the garage. On the other

hand, there were some limitations with this approach such as the inability to tweak the behaviour because there are no parameters to update. Also, after some time, the oscillations tend to grow leading it to diverge which makes this approach not the best one.

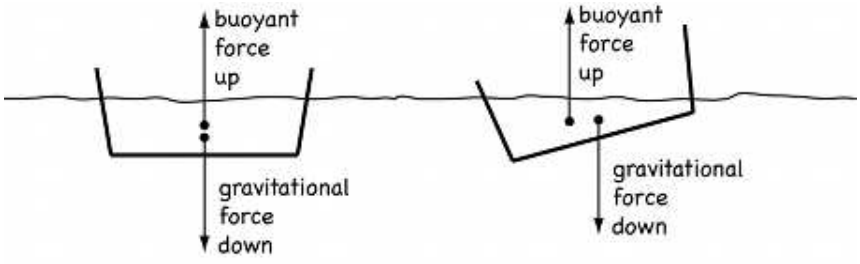


Figure 5.7: Rectangular shaped hull [10]

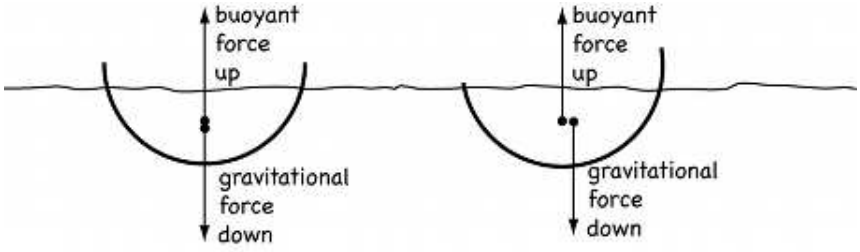


Figure 5.8: Semicircular shaped hull [10]

5.1.3 Writing a custom plugin

Another approach was implemented to simulate the disturbances on the docking garage but unlike the other approaches 5.1.1 and 5.1.2, this approach works by applying the forces and torques directly on the docking garage instead of applying them on the surface vessel and let them transmit to the garage via the cable. To implement this approach, a custom plugin was developed based on the Entity Component Manager(ECM) software architecture to apply virtual forces and torques on the docking garage directly. These virtual forces and torques will simulate realistic disturbances. One approach to model these forces and torques is to implement a sine wave where the frequency and amplitude could be modified to match the desired behaviour. Another approach is to implement a sum of sines to have more realistic behaviour. Figure 5.9 illustrates the equations used in modelling the forces along the three axes as well as the torque around the z-axis. Figure 5.10 illustrates the parameters required by the custom plugin. These parameters will be updated in the SDF file and will update the amplitudes and frequencies values in the modelling equations as shown in Figure 5.9. The approach of using a custom plugin to simulate the disturbances on the docking garage was so far the best approach

because you can update and tweak the parameters to reach the desired behaviour. It also doesn't conflict with the other plugins in the simulation.

```
  double x_force = myParameters["amplitude_xf"] * sin(2 * PI * myParameters["frequency_xf"] * t_);

  double y_force = myParameters["amplitude_yf"] * sin(2 * PI * myParameters["frequency_yf"] * t_);

  double z_force = myParameters["amplitude_zf"] * sin(2 * PI * myParameters["frequency_zf"] * t_);

  double z_torque = myParameters["amplitude_zt"] * sin(2 * PI * myParameters["frequency_zt"] * t_ *
exp(myParameters["D_zt"]*t_));
```

Figure 5.9: Equations used to model the forces and torques in the developed custom plugin

```
<plugin filename="libDockingStation.so" name="docking_station::DockingStation">
<frequency_xf>0.01</frequency_xf>
<frequency_yf>0.01</frequency_yf>
<frequency_zf>0.01</frequency_zf>
<frequency_zt>0.2</frequency_zt>

<amplitude_xf>0.0025</amplitude_xf>
<amplitude_yf>0.0025</amplitude_yf>
<amplitude_zf>25</amplitude_zf>
<amplitude_zt>1.75</amplitude_zt>

<D_zt>0.01</D_zt>

</plugin>
```

Figure 5.10: The parameters needed to model the forces and torques in the developed custom plugin

5.2 Homing - Garage detection

Regarding autonomous docking and as explained before, there are two main elements: homing and docking. Homing is about searching for the garage and getting close to the garage while docking is when the robot is close to the garage and about to dock. In this section, the implementation of the garage detection algorithm as part of the homing algorithm will be discussed. The motivation behind the garage detection algorithm is to be able to recognize the garage from the camera stream. In addition to garage recognition, the robot will be able to track the garage based on the garage detection information. To achieve that target they were mainly two general approaches:

Traditional object detection:
There are several traditional approaches that can be used to do the object detection tasks without depending on AI such as Haar Cascades, Histogram of Oriented Gradients (HOG), Edge-based Approaches, Scale-Invariant Feature Transform (SIFT), and Template Matching. Despite being relatively computationally efficient and interpretable, these approaches lack good generalization, adaptability and robustness to variability.
Deep learning-based approaches:
Nowadays, there is a lot of development in Deep learning-based models for computer vision tasks such as object detection, object tracking, segmentation, etc. One of the popular methods is YOLO which stands for You Only Look Once. YOLO is known for its efficiency and accuracy in object detection tasks in images and videos. At its core, YOLO solves the object detection tasks as a regression problem, where it directly predicts the bounding boxes and class probabilities for objects in a single pass over the input image. While such approaches proved to be highly efficient, they may need a good amount of training dataset to perform well.

After studying the general approaches in object detection, the YOLOv5 was chosen. To achieve the garage detection task using YOLOv5, several steps were implemented:
Generating datasets:
The very first step was generating datasets for training the YOLO. In the beginning, Within the context of this task which is to detect the garage in the image plane, it was assumed that there will not be any other objects in the scene other than the garage. In other words, the only object that the robot will see while doing the docking would be the docking garage unless there is a nearby underwater structure or a underwater animal which is a rare scenario to happen so this assumption is valid. To generate an appropriate dataset for the YOLO, there is a simple format to follow as shown in Figure 5.11.
To generate the data set in this format, there are mainly two ways:
Manually annotating the dataset using software platforms:
There are many platforms to help you manually annotate your data. You can use trainyolo.com but this platform is end to end approach and you will have your trained model afterwards. It can take between 10-20 seconds to annotate each image.
Automatically generating and annotating the dataset:
This approach is only valid for generating datasets in the simulation. You can utilise the boundingbox camera plugin in Gazebo. To use this plugin, you first need to add the ignition-gazebo-label-system plugin to your link which in our case is the garage. This approach was used to generate the training dataset from the simulation. However, later in training the yolo on real images, the manual annotation would be our only way which of course would take more time.
Training and testing: To train the yolov5, the yolov5 repository from Ultralytics was used [3]. Ultralytics is one of the leading companies in cutting-edge AI; it specializes in making AI models easier to use. So, by using their yolov5 repo, there would not be a need to start training the network from scratch. It is just required to organize the directory containing the training dataset in the correct format which is illustrated in

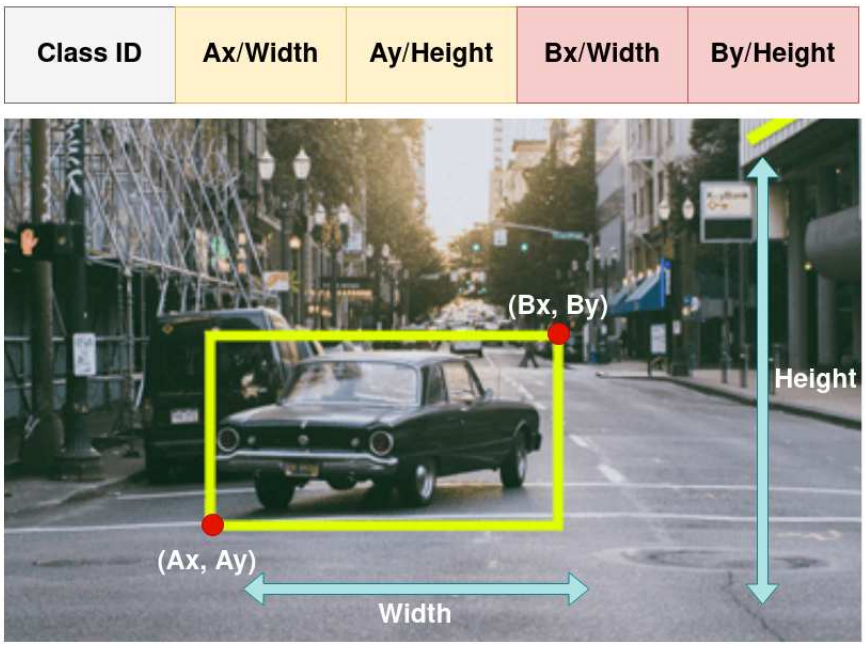


Figure 5.11: The training dataset format for the yolo algorithm

Figure 5.12. And then, a Python script could be run to start the training after specifying some hyper-parameters such as the number of epochs and batch size.

```
├── garage_dataset/:
|   ├── dataset.yaml
|   ├── images/:
|   |   └── train/: All images files for training
|   |   └── val/: All images files for validation
|   ├── labels/:
|   |   └── train/: All txt labels files for training
|   |   └── val/: All txt labels files for validation
```

Figure 5.12: How the directory containing the dataset should be organized in order for Ultralytics's script to work properly

Integrating YOLO with ROS The final step after training and testing the yolov5 model for the garage detection task is to integrate with ROS to work alongside the other algorithms. So a new ros2 node was created that will load the trained yolov5 model once loaded. The node will keep subscribing to the ros2 topic containing the image stream and for each input image the yolov5 with output the detection information as well as a new image stream containing a visualization of the detected garage. Figure 5.13 illustrates the ros2 node that integrates the yolov5 model.

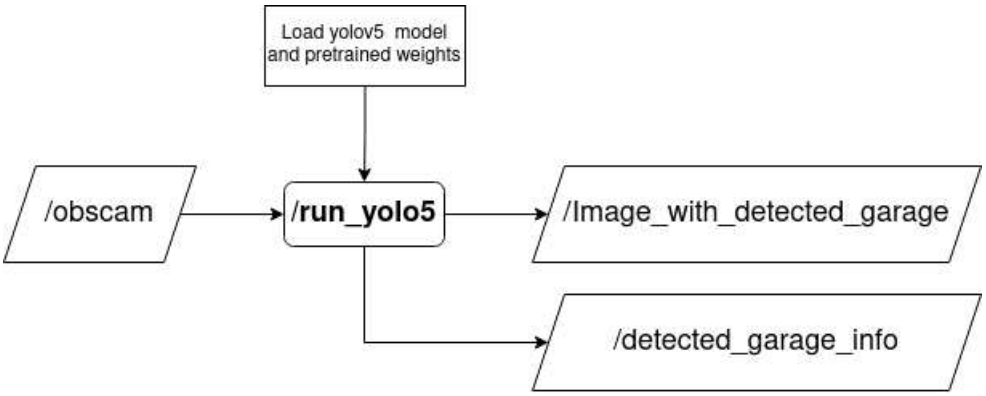


Figure 5.13: The ros node for running yolov5

5.3 Homing - Control strategy

After having the garage detection algorithm working, the next step would be to implement the control strategy to find the garage and do the homing. The core idea behind this strategy is to search for the garage and recognize it using the yolov5 and then approach it and face the side of the markers. By doing the last step, the homing would be completed and the docking would be the next and final phase. Figure 5.14 summarizes the proposed control strategy for the homing.

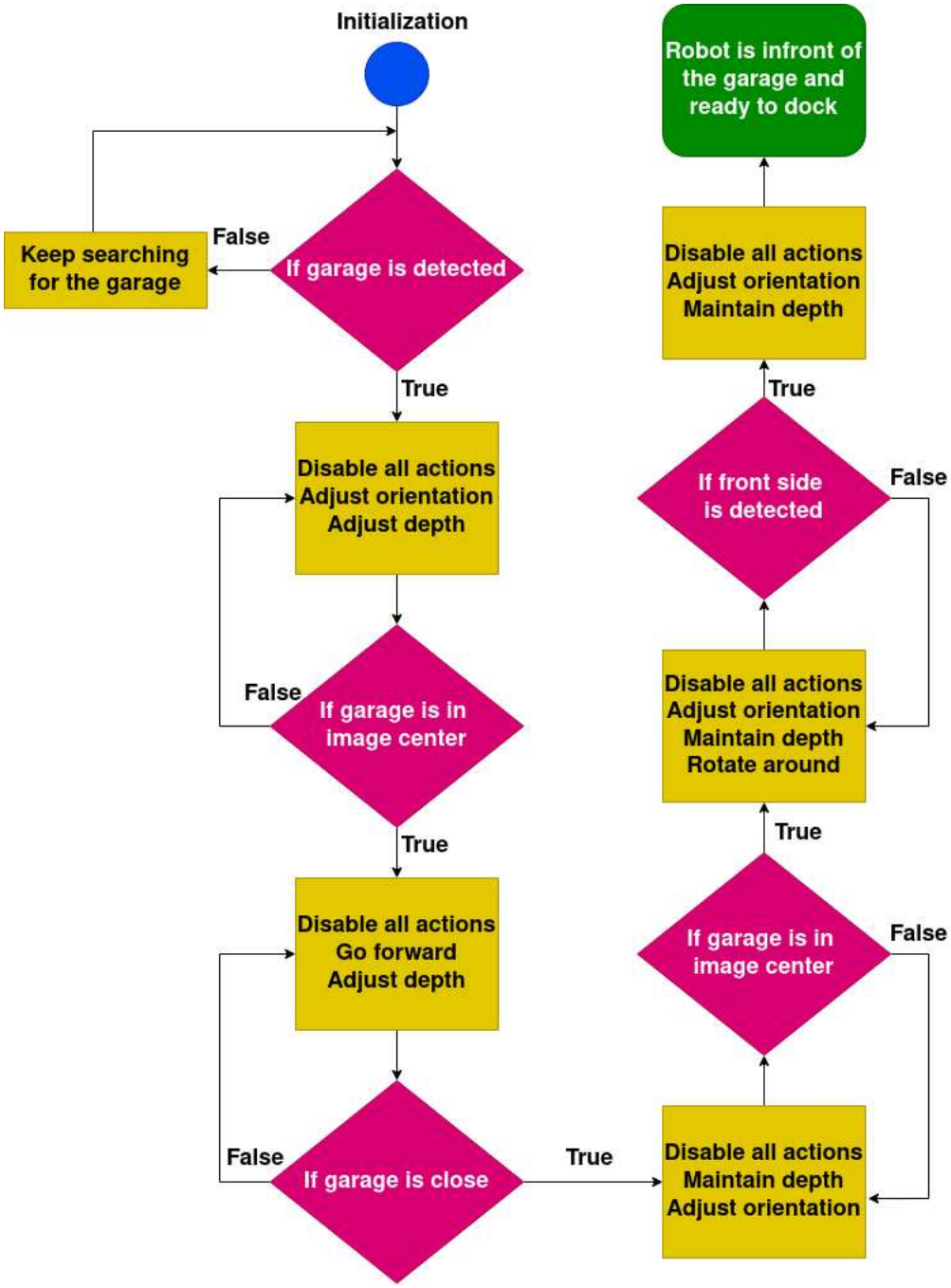


Figure 5.14: Proposed control strategy for the homing

5.4 Docking - Motion estimation

This section will discuss the implementation of the motion estimation algorithm and the different approaches taken. As explained in the previous chapters, developing a motion estimation algorithm is important because the oscillations of the docking garage affect the detection frequency of the markers. Also by having a good motion estimation model, it would be possible to predict the garage states in the future time steps which will allow for potential path planning.

5.4.1 Linear model - Kalman Filter

In the beginning, a Kalman filter with a linear model was implemented to investigate its performance. The aim of the linear Kalman filter implemented in this approach is to estimate the position and linear speeds. On the other hand, we can only measure the position when the camera detects the markers on the docking garage. The step for calculating the garage's pose through the markers has been already implemented by the team at Forssea Robotics company. Equation 5.1 illustrates the states and measurements in our linear Kalman filer model. The Equations below illustrate the motion equations that will be later implemented in the form of a matrix. Algorithm 5.4.1 summarizes the conventional implementation of Kalman filter where the are two main steps: prediction and update. Prediction is when the Kalman filter estimates the garage's position based on the motion model only. While for the update phase, the Kalman filter will use the measurements to obtain a better estimation of the garage's position.

$$X_k = \begin{bmatrix} x_k \\ y_k \\ z_k \\ u_k \\ v_k \\ w_k \end{bmatrix}, Z_h = \begin{bmatrix} x_h \\ y_h \\ z_h \end{bmatrix} \tag{5.1}$$

$$x_{k+1} = x_k + u_k \Delta t + \frac{1}{2} a \Delta t^2 \tag{5.2}$$

$$\mathrm{y}_{k+1} = y_k + v_k \Delta t + \tfrac{1}{2} a \Delta t^2 \tag{5.3}$$

$$\mathrm{z}_{k+1} = z_k + w_k \Delta t + \tfrac{1}{2} a \Delta t^2 \tag{5.4}$$

$$\mathrm{u}_{k+1} = u_k + a \Delta t \tag{5.5}$$

$$\mathrm{v}_{k+1} = v_k + a\Delta t \quad (5.6)$$

$$\mathrm{w}_{k+1} = w_k + a\Delta t \quad (5.7)$$

$$\mathrm{X}_{k+1} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{F} X_k + \underbrace{\begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 \\ \Delta t \\ \Delta t \\ \Delta t \end{bmatrix}}_{G} a \quad (5.8)$$

Where $a \sim N(0, \sigma^2)$

$$\mathrm{Z}_k = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}}_{H} X_k \quad (5.9)$$

Algorithm 1 The implementation of the Kalman filter

Require: *subscribtion to garage detection and markers detection topics*

while *Running* **do**

$Predict:$

$x \leftarrow Fx$

$P \leftarrow FPF^t + GG^t a$

$Update:$

$y \leftarrow z - Hx$

$S \leftarrow HPH^t + R$

$K \leftarrow PH^t S^{-1}$

$x \leftarrow x + Ky$

$P \leftarrow (I - KH)P$

end while

5.4.2 Non-linear model - Extended Kalman Filter

Another approach that was implemented to estimate the garage's position was considering the garage as a 3D pendulum model with an elastic link "representing the heave oscillations". The following figure illustrates the modelling of a spherical pendulum with mass m and vertical and horizontal angles. Given this modelling, our equations for the position as well as the linear speed will be as shown in the equations below.

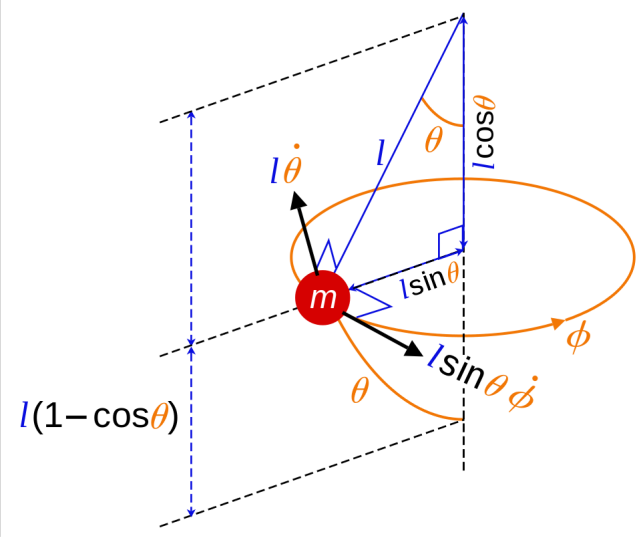


Figure 5.15: 3D Pendulum angles and velocities [9]

$$x = lsin(\theta)cos(\phi) \tag{5.10}$$

$$y = lsin(\theta)sin(\phi) \tag{5.11}$$

$$z = l_0 - L(t)cos(\theta) \tag{5.12}$$

but since, in real life, the docking garage should experience small changes in orientation, we can, then, expect small changes in θ and ϕ, so we can use the small angle approximation and assume $sin(\theta)$ and $cos(\theta)$ to be θ and 1 respectively and the same applies to ϕ as well. So the simplified equations will be:

$$x = l\theta \tag{5.13}$$

$$y = l\theta\phi \tag{5.14}$$

$$z = V(t) \tag{5.15}$$

Moreover, in our scenario, we have high oscillations along the z-direction so the length of the pendulum is not fixed. To simulate the vertical oscillations, let's assume a sinusoidal function V, where:

$$V(t) = a_z sin(2\pi f_z t) \tag{5.16}$$

$$L(t) = l + V(t) \tag{5.17}$$

Where a_z and f_z are the amplitude and the frequency of the waves respectively. And regarding θ and ϕ, we can also assume their values to follow sinusoidal patterns, but with very small amplitudes and frequencies. So we can assume the following:

$$\theta(t) = a_\theta sin(2\pi f_\theta t) \tag{5.18}$$

$$\phi(t) = a_\phi sin(2\pi f_\phi t) \tag{5.19}$$

So we will modify equations 5.13 to 5.15 to be:

$$x = L(t)\theta(t) \tag{5.20}$$

$$y = L(t)\theta(t)\phi(t) \tag{5.21}$$

$$z = L(t) \tag{5.22}$$

Or in more detailed representation:

$$x = [l + a_z sin(2\pi f_z t)][a_\theta sin(2\pi f_\theta t)] \tag{5.23}$$

$$y = [l + a_z sin(2\pi f_z t)][a_\theta sin(2\pi f_\theta t)][a_\phi sin(2\pi f_\phi t)] \tag{5.24}$$

$$z = [l + a_z sin(2\pi f_z t)] \tag{5.25}$$

So regarding the velocities, we can take the first-order derivatives and obtain this

$$\dot{x} = [2\pi f_z a_z cos(2\pi f_z t)][a_\theta sin(2\pi f_\theta t)] + [l + a_z sin(2\pi f_z t)][2\pi f_\theta a_\theta cos(2\pi f_\theta t)] \tag{5.26}$$

$$\begin{aligned}\dot{y} = {} & [2\pi f_z a_z cos(2\pi f_z t)][a_\theta sin(2\pi f_\theta t)][a_\phi sin(2\pi f_\phi t)] \\ & + [\mathrm{l}+\mathrm{a}_z sin(2\pi f_z t)][2\pi f_\theta a_\theta cos(2\pi f_\theta t)][a_\phi sin(2\pi f_\phi t)] \\ & + [\mathrm{l}+\mathrm{a}_z sin(2\pi f_z t)][a_\theta sin(2\pi f_\theta t)][2\pi f_\phi a_\phi cos(2\pi f_\phi t)]-\end{aligned} \tag{5.27}$$

$$\dot{z} = 2\pi f_z a_z cos(2\pi f_z t) \tag{5.28}$$

Prediction step So regarding our state vector, we can assume the following:

$$\vec{X}_{t+1} = f(x,u) = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \\ \dot{x}_{t+1} \\ \dot{y}_{t+1} \\ \dot{z}_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + \dot{x}_t t \\ y_t + \dot{y}_t t \\ z_t + \dot{z}_t t \\ \dot{x}_{t+1} \\ \dot{y}_{t+1} \\ \dot{z}_{t+1} \end{bmatrix} = \begin{bmatrix} X[0] + X[3]\Delta t \\ X[1] + X[4]\Delta t \\ X[2] + X[5]\Delta t \\ \dot{x} \text{ ”5.26”} \\ \dot{y} \text{ ”27”} \\ \dot{z} \text{ ”5.28”} \end{bmatrix} \tag{5.29}$$

$$F = \frac{\partial f(x,u)}{\partial x}\Big|_{x_t, u_t} \tag{5.30}$$

$$F = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.31}$$

$$P = FPF^T + Q \tag{5.32}$$

Update step

$$y = z - H\vec{X} \tag{5.33}$$

$$K = PH^T(HPH + R)^{-1} \tag{5.34}$$

$$\vec{X} = \vec{X} + Ky \tag{5.35}$$

$$P = (I - KH)P \tag{5.36}$$

5.4.3 Non-linear model - Extended Kalman Filter with learning

The previous approach of using an extended Kalman filter didn't show good results. I believe the reason is that the proposed motion model based on 3D pendulum modelling was too complex and not accurate. Motivated by that and based on the available data on the garage's movements, I implemented another approach that assumes the garage's movements along the 3 axes to follow a sinusoidal pattern. The only problem was which parameters (amplitude, frequency, phase shift) to use in the motion model to make the EKF able to predict the garage's position. To solve this issue, a step for learning and updating these parameters was implemented. The core idea is that the Argos ROV will approach the garage and observe the markers and in turn calculate its position. After collecting enough data, the algorithm will be able to extract the desired parameters and update the motion model in the prediction step. Calculating enough data can be

determined by the duration of the dataset or the number of points collected. Figure 5.16 summarizes the implemented approach where the EKF will initialize as normal and if there are no measurements, the EKF will estimate the garage's position via prediction only. While if the robot could detect the markers in the garage and calculate the pose, the algorithm will keep recording the measurements until it collects enough data for learning. Then the algorithm will run the complete EKF (prediction + update) as well as learn the parameters and update the motion model.

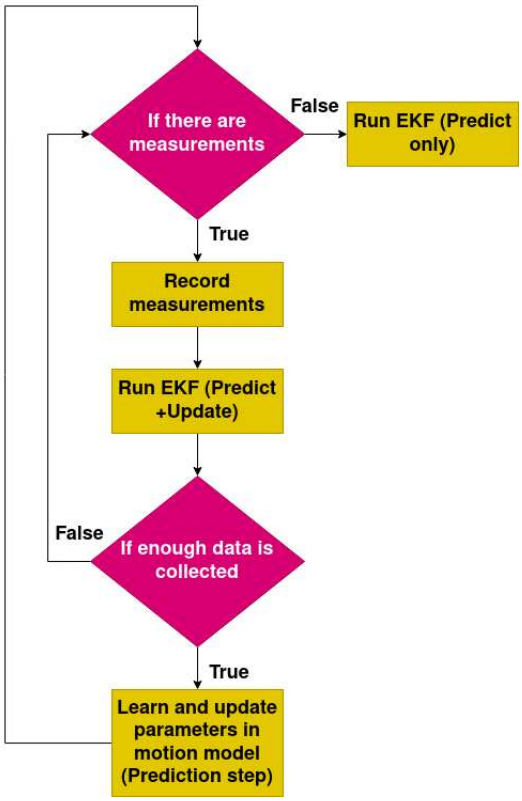


Figure 5.16: Proposed learning-based EKF approach

Figure 5.17 demonstrates which parameters to learn. There are two models implemented: one sine model and three sines model. In the first model, the measurement signal is assumed to follow a simple sinusoidal pattern meaning that there are only 3 parameters to estimate for each measurement signal. While in the three-sines model, each measurement signal is assumed to follow a complex sinusoidal pattern of a sum of three sines; hence 9 parameters are to be learned for each signal. Figure 5.18 explains how to obtain the desired parameters to update the motion model in EKF where first the measurement signal will pass by a Fast Fourier Transform (FFT) to extract the signal components represented by amplitudes and frequencies. The implemented FFT algorithm is one of the functions available in scipy library in Python. Then these values will be used as

initial guesses for the curve fitting function to have a final estimation of the amplitudes, frequencies, and phase shifts. The curve fitting function implemented here is one of the functions in scipy optimization library in Python and it works better when realistic initial guesses of the amplitudes and frequencies are provided; that is why integrating it with FFT was a good idea. Figure 5.19 illustrates how the FFT works which is basically decomposing the input signal to its component signal represented by amplitudes and frequencies.

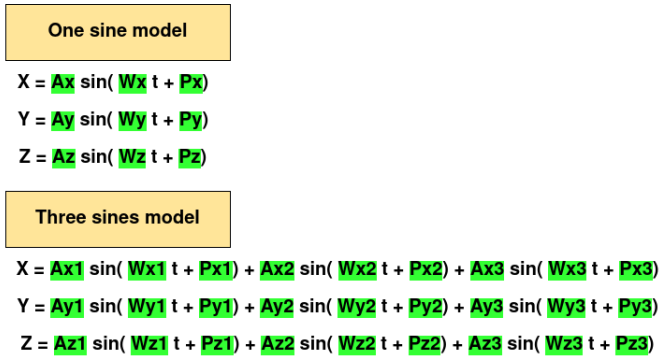


Figure 5.17: Which parameters to update in the motion model for both sine and 3 sines models

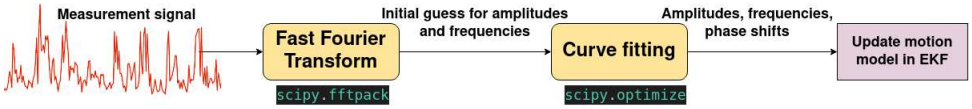


Figure 5.18: How to obtain the desired parameters in Figure 5.17 using Fourier transform and curve fitting

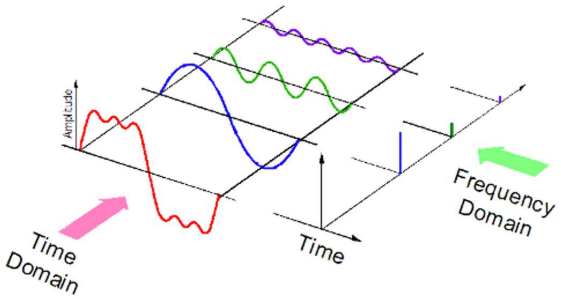


Figure 5.19: How Fast Fourier Transform (FFT) works [15]

5.5 Collecting datasets from the real garage

The main information of interest was the garage's movements as well as images from different orientations of the garage. To do so an ISD 4000 sensor was attached to the garage [6]. ISD 4000 has an imu, pressure, and temperature sensor. From the imu, we can easily get the pitch, roll, and yaw orientations. And using the pressure readings, we can calculate the depth. Figure 5.20 shows the ISD 4000 sensor that was attached to the garage's frame. In an ideal world, I would be interested in horizontal movements as well to further develop the simulation. Regarding collecting images, the Argos ROV already has a camera onboard. Figure 5.21 summarizes which data was collected during the trial at Sète. Figure 5.22 shows the garage and Argos on the vessel before deploying them in water. Figures 5.23 and 5.24 show the Argos and the garage after dropping them into water respectively.

Figure 5.20: ISD4000 sensor that was used to record garage orientation and depth [6]

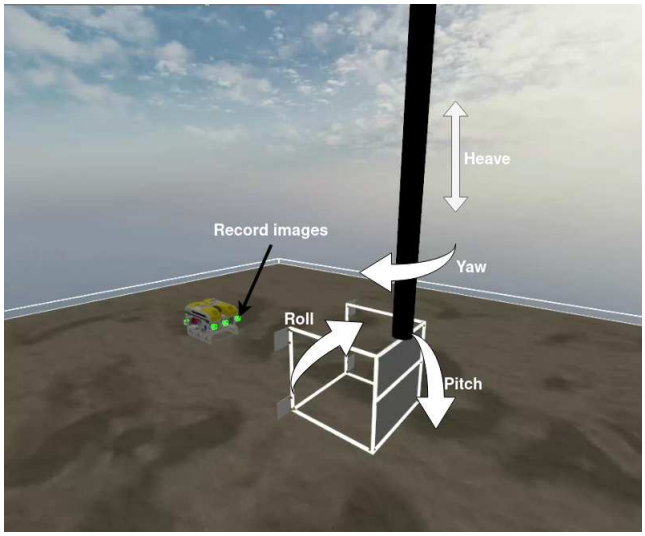


Figure 5.21: Which information to record during the trial at Sète

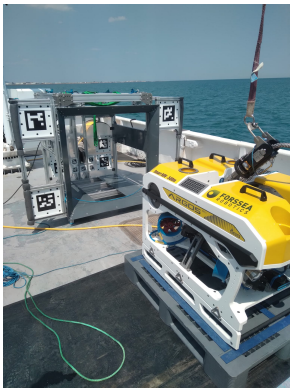


Figure 5.22: Argos and the docking garage onboard

Figure 5.23: Argos in water

Figure 5.24: Dropping the garage in water

Chapter 6

Results

If you can't measure it, you can't improve it - Lord Kelvin

Contents

6.1 Gazebo simulation of docking garage **51**
6.2 Motion prediction **58**
6.3 Garage detection **64**
6.4 Collecting real dataset **65**

This chapter will discuss results and comparisons between different approaches with each milestone. The content of this section will lay out the ground for making a conclusion and future work in chapter 7.

6.1 Gazebo simulation of docking garage

This section will present the results of the different approaches to building the simulation in Gazebo as explained in section 5.1. Building the simulation for underwater docking was important in developing the solutions later as it will save a lot of time and resources. The core of the new simulation is to simulate the disturbances acting on the docking garage and by turn have a realistic garage behaviour that will allow for testing the autonomous docking algorithm later. As explained in Chapter 5, three approaches have been implemented to simulate the garage's movement.

Figures 6.1 and 6.2 illustrate the garage behaviour along the x and y axes in the first approach which was using wave sim plugin. It is clear that the horizontal oscillations were aggressive with high amplitudes and frequencies. On the other hand, the oscillations

along the z-axis have more realistic behaviour. The amplitudes of the oscillations along the z-axis were on average 0.35m which is realistic in some scenarios as shown in Figure 6.3. Moreover, the heading oscillations were logical and expected as demonstrated in Figure 6.4 where the garage was changing its orientation gradually as a result of the cable tension.

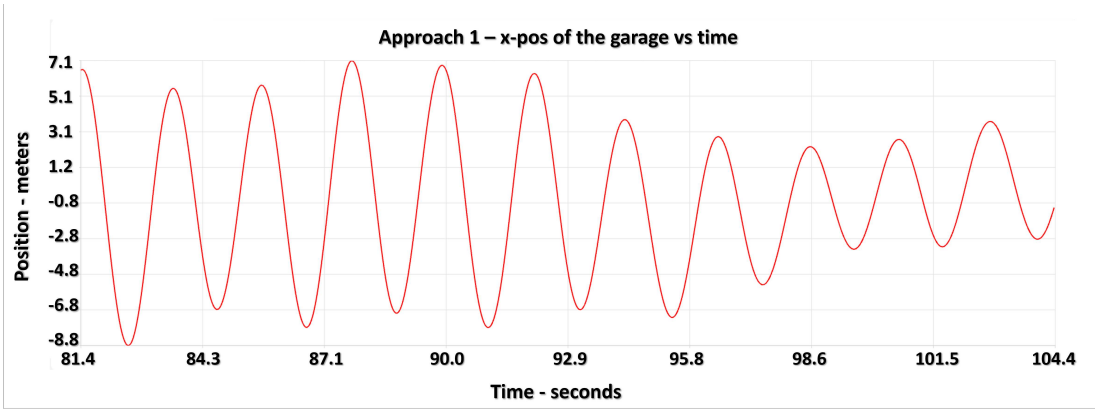


Figure 6.1: Approach 1 - x-position of the garage

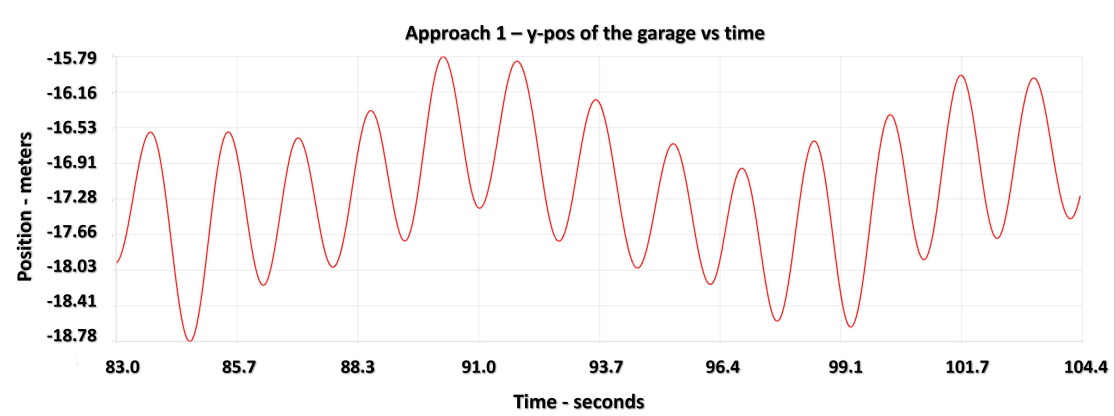


Figure 6.2: Approach 1 - y-position of the garage

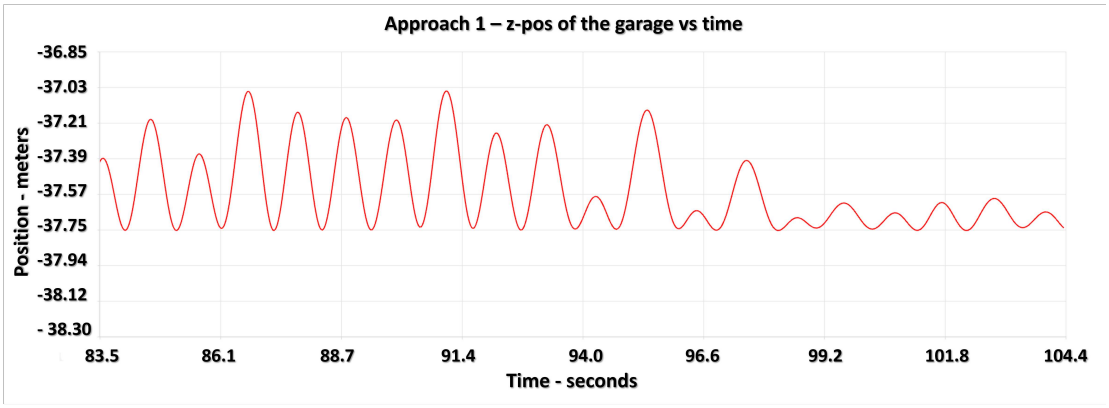


Figure 6.3: Approach 1 - z-position of the garage

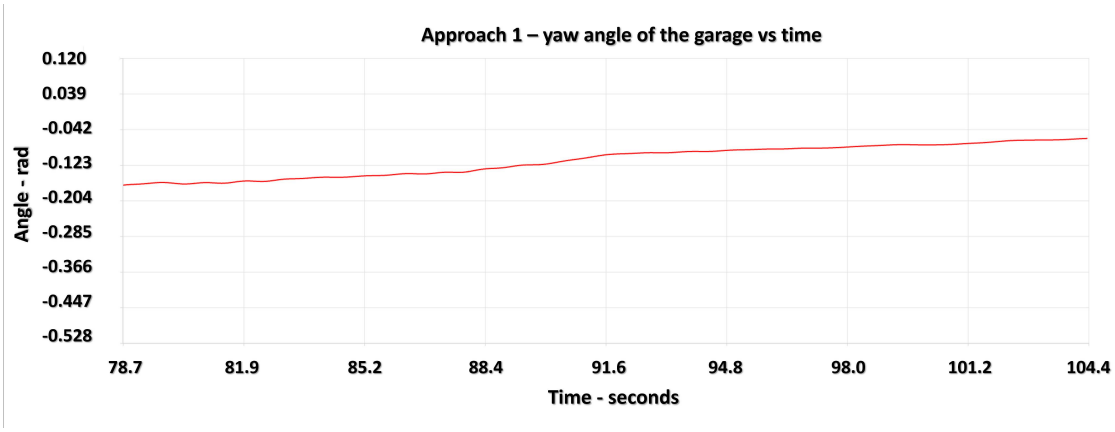


Figure 6.4: Approach 1 - yaw-angle of the garage

Moving to approach 2 when the buoyancy plugin was only used as explained in chapter 5.1.2, there was high amplitude along the x-direction which is unrealistic as shown in figure 6.5. Moreover, the oscillations along the y-axis were very small which may be true in some sea states as illustrated in Figure 6.6. On the other hand, Figures 6.7 and 6.8 demonstrate very realistic oscillations along the z-axis as well as the garage heading respectively.

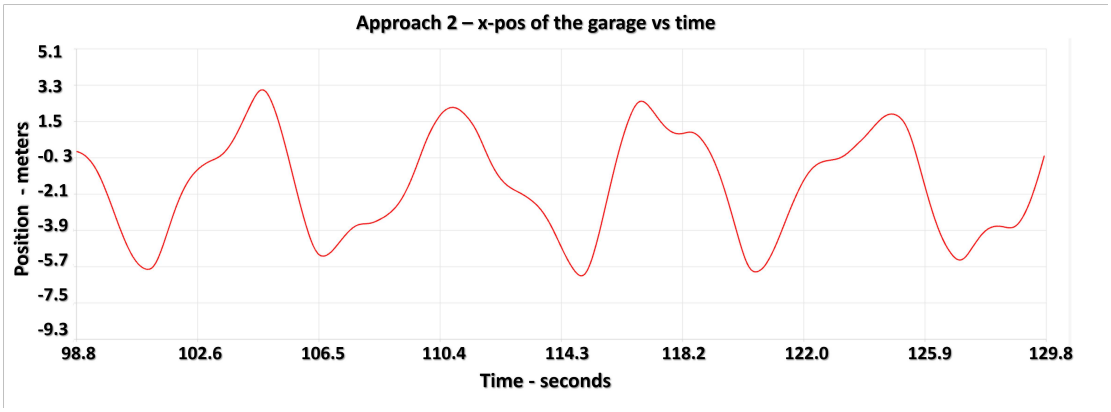


Figure 6.5: Approach 2 - x-position of the garage

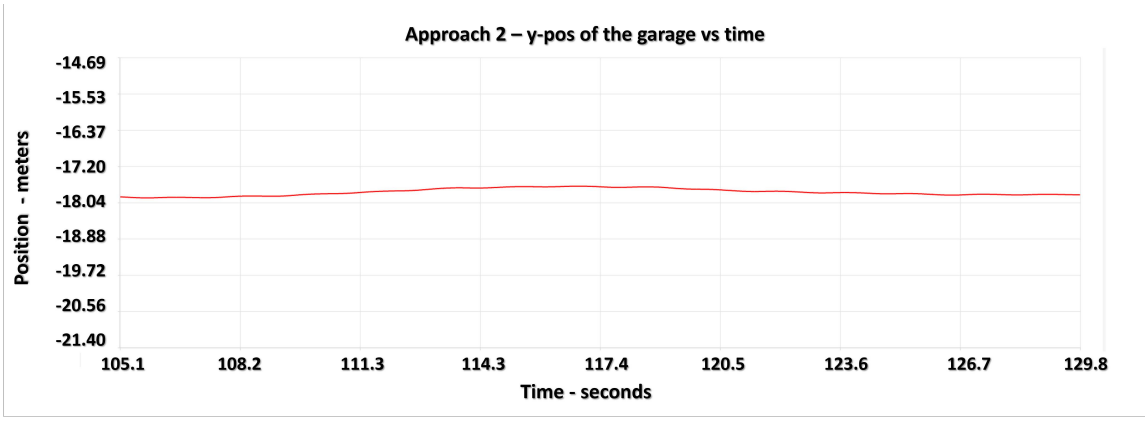


Figure 6.6: Approach 2 - y-position of the garage

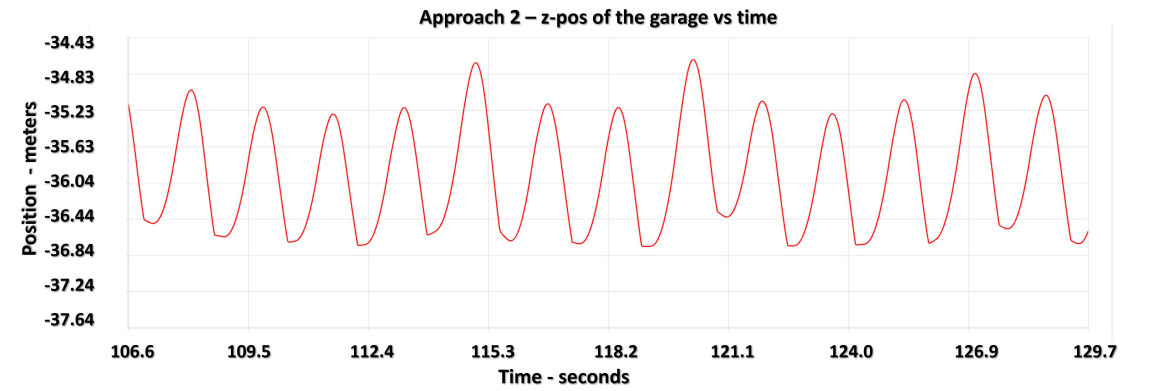


Figure 6.7: Approach 2 - z-position of the garage

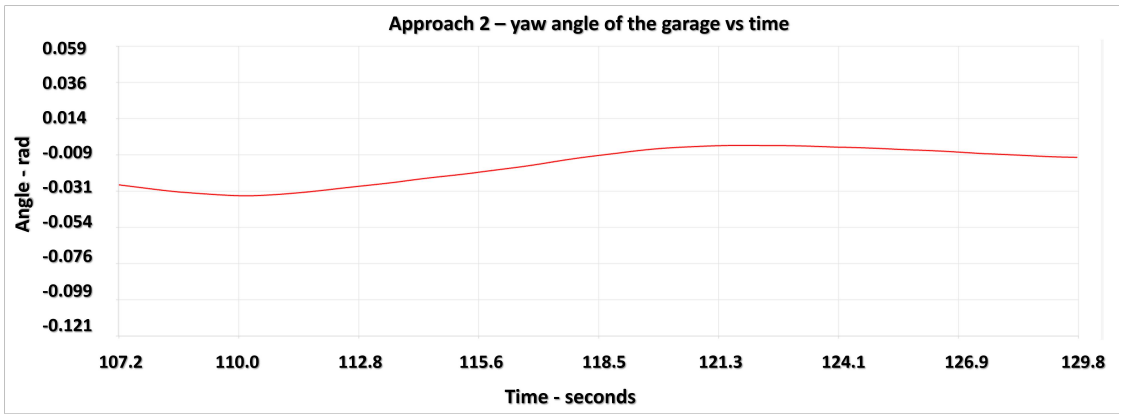


Figure 6.8: Approach 2 - yaw angle of the garage

In the third approach of simulating the docking garage, a custom gazebo plugin was developed to simulate the disturbances as previously explained in section 5.1.3. Figures 6.9 and 6.10 show the simulated garage's behaviour along the x and y axes respectively. It is clear that both the amplitude and frequency values were low which is a realistic pattern given the information from the literature and the previous team experience. Furthermore, Figures 6.11 and 6.12 illustrate the garage's movement along and around the z-axis respectively. It is obvious that the wave amplitude is small which is true in the case of calm sea state conditions. On the contrary, the heading oscillations were relatively high in terms of amplitude and frequency which still can be true in some sea state conditions.

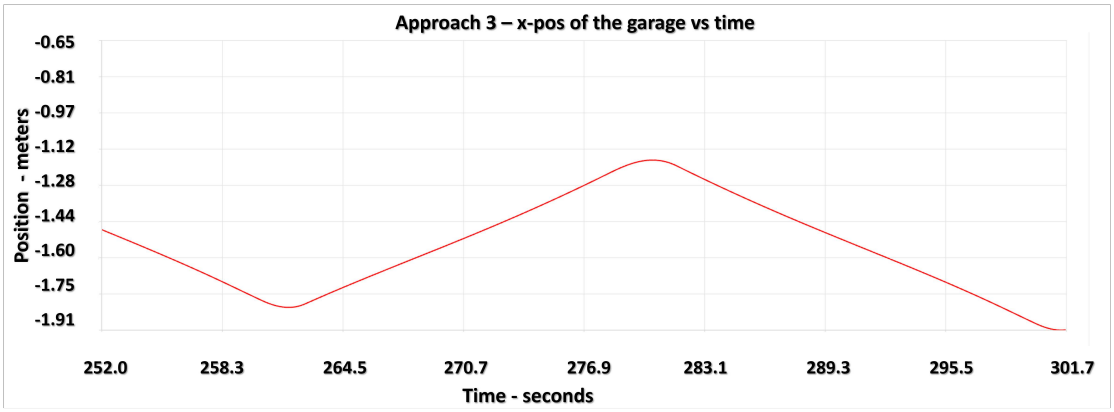


Figure 6.9: Approach 3 - x-position of the garage

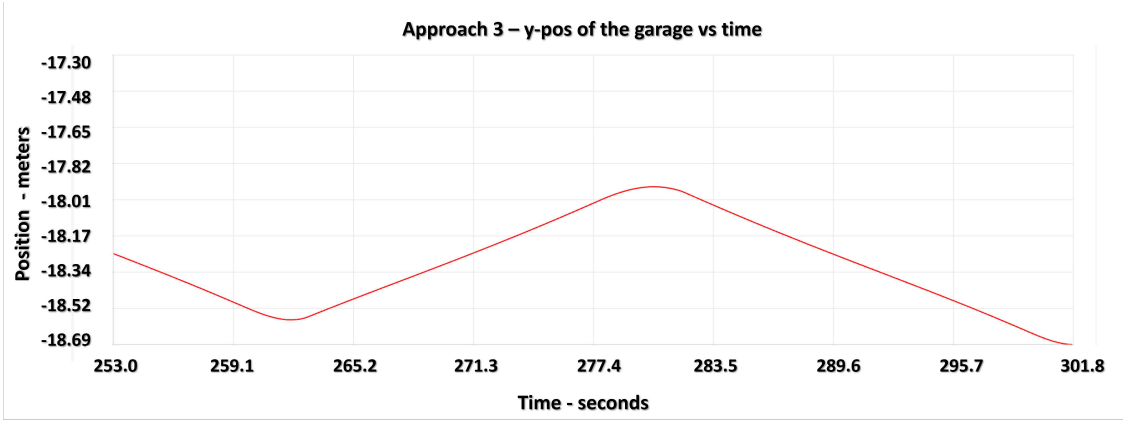


Figure 6.10: Approach 3 - y-position of the garage

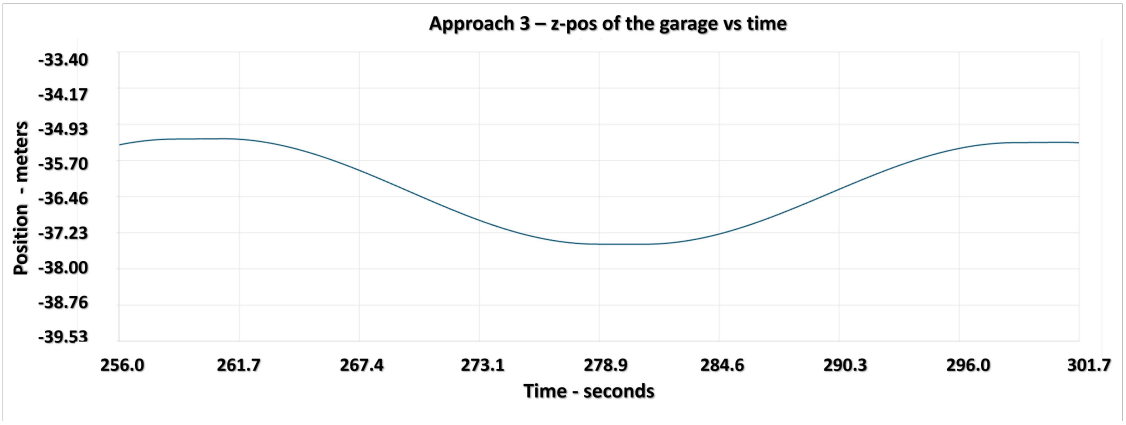


Figure 6.11: Approach 3 - z-position of the garage

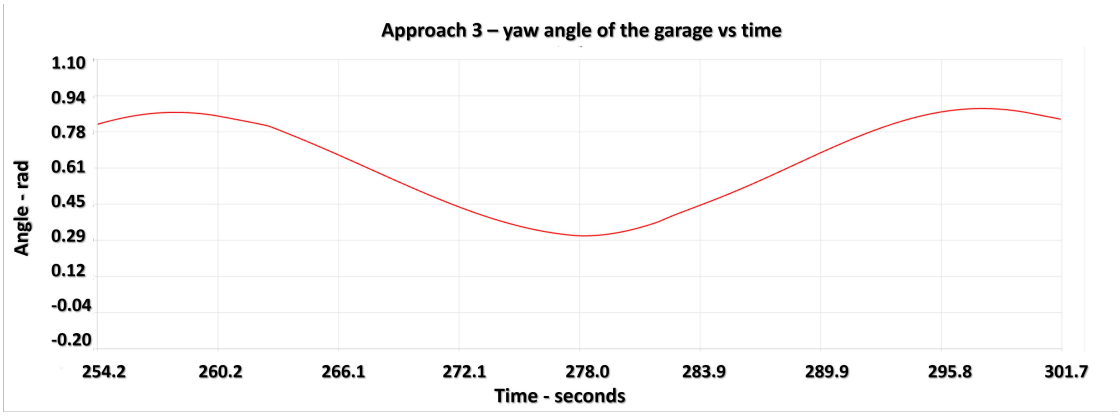


Figure 6.12: Approach 3 - yaw angle of the garage

6.2 Motion prediction

This section will discuss the results of the different approaches taken in developing a motion estimation algorithm for the docking garage as explained in section 5.4.

6.2.1 Linear model - Kalman Filter

In the first approach, a linear motion model was implemented in the Kalman filter. Since there was no available dataset from the docking garage, proposed sine waves with different amplitudes and frequencies were used to simulate the garage's movement directly and artificial noise was added to make them more realistic. Figures 6.13; 6.14, 6.15 illustrate the performance of the proposed Kalman filter with linear motion model with different update rate: 5hz, 1hz, 0.5hz respectively. It is clear that the higher the update frequency, the better the KF output which is reasonable. If a high and consistent update rate is guaranteed then using this approach may be a good solution. However as explained before in section 4.1, one of the challenges is that the visibility may be bad and the markers detection algorithm may not be able to calculate the garage pose. To test our motion estimation algorithm given this challenge, a new test was conducted where there was a time period of no update representing a loss of detection or inability to calculate the garage pose. Figure 6.16 shows an example of such simulation where the red area represents no measurements and in turns no update. It is obvious that the KF output was far from the true value and this is because the implemented motion model was linear and the garage movement follow a non-linear pattern.

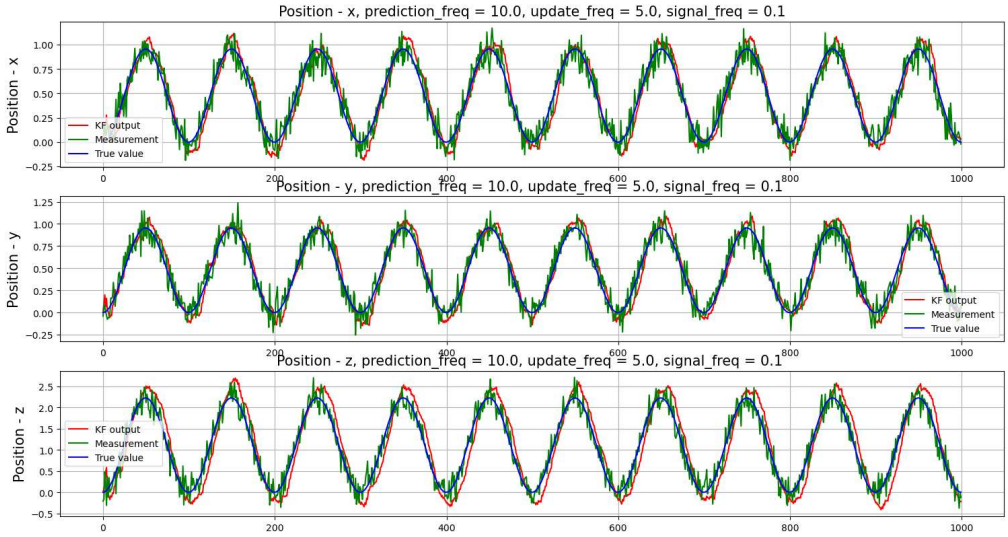


Figure 6.13: KF - prediction freq: 10hz - update freq: 5hz

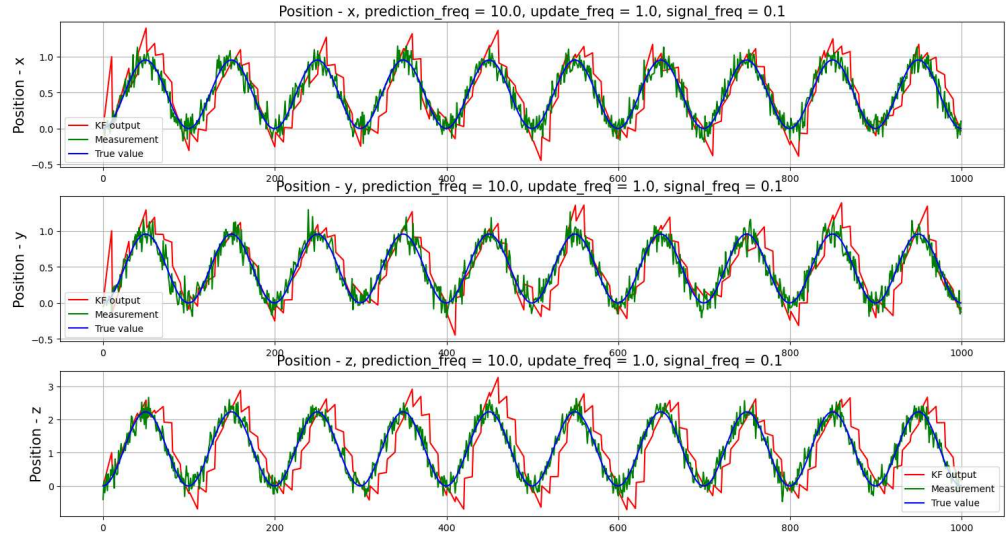


Figure 6.14: KF - prediction freq: 10hz - update freq: 1hz

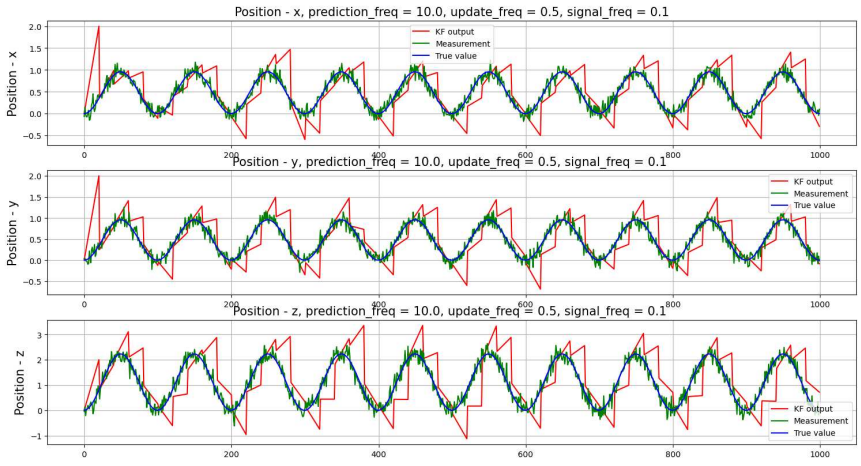


Figure 6.15: KF - prediction freq: 10hz - update freq: 0.5hz

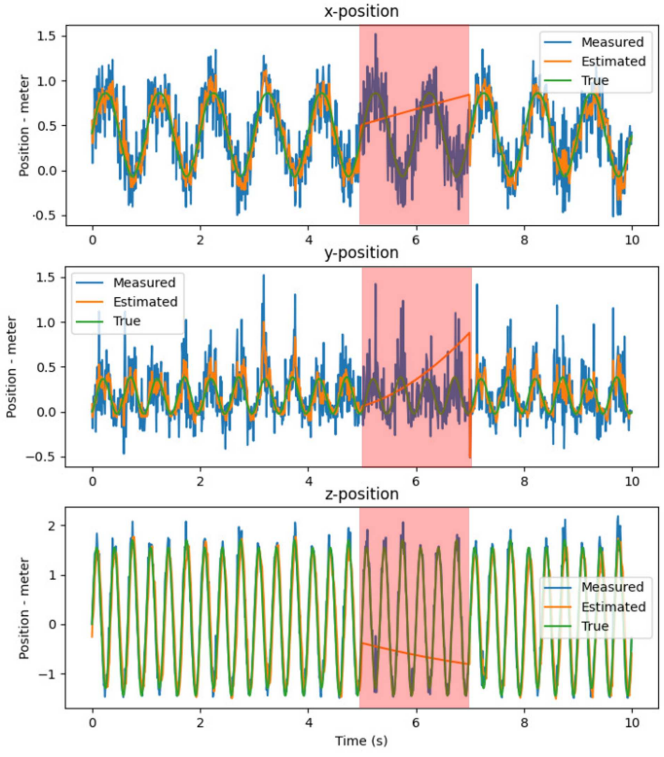


Figure 6.16: An example of when KF can fail if it didn't update for few seconds as shown in the red area

6.2.2 Extended Kalman filter with learning

Since the Kalman filter with linear motion model showed poor estimation in case of detection loss, developing an extended Kalman filter was the logical thing to do. When a non-linear motion model based on 3D pendulum equations was implemented, the EKF still failed to output good estimation in case of bad detection. The main reason behind this is the complexity of modelling the docking garage as a 3D pendulum. The model was too complex with many assumptions that led at the end to have an incorrect motion model. To tackle this issue, another approach was implemented assuming that the garage movements are following a sine pattern or even sum of sines pattern. The only issue with this approach is that we don't know the parameters of these patterns such as amplitude and frequency. That's why a learning phase has been implemented which works by first observing the garage movement for some time period and then estimating the amplitudes and frequencies based on Fourier transform. Figure 6.17 shows what would be the EKF performance if did not update nor learn. It is clear that without any learning or updating, the motion model alone could not give a good estimation of the garage's position. Moreover, Figure 6.18 shows the EKF's performance when it learns the garage performance for 4 seconds at the beginning and then uses the learned parameters in updating the motion model. As we see the EKF was able to output good estimation despite not updating. The EKF's performance was better in estimating the z position because this one had a higher frequency resulting in more cycles to be used in learning. Figure 6.19 is similar to 6.18 but the update function was turned on. The EKF's performance, when it combined learning and updating, demonstrated a much better estimation of the garage's position.

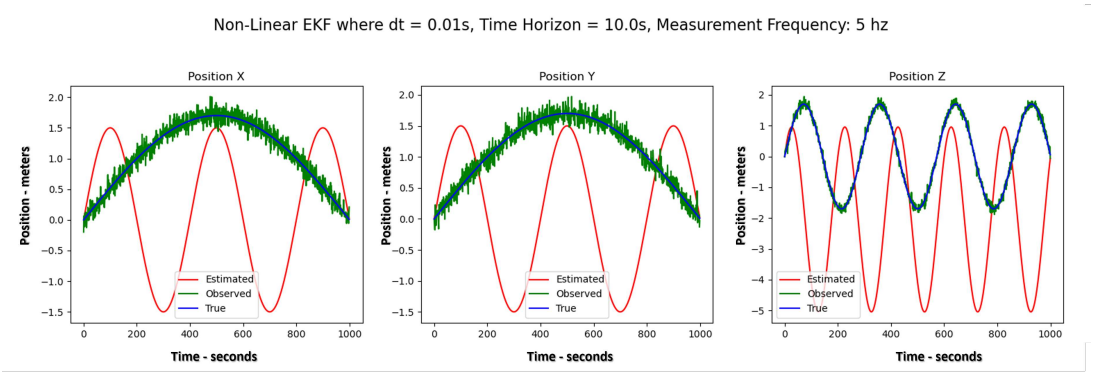


Figure 6.17: EKF without update and without learning - offline testing

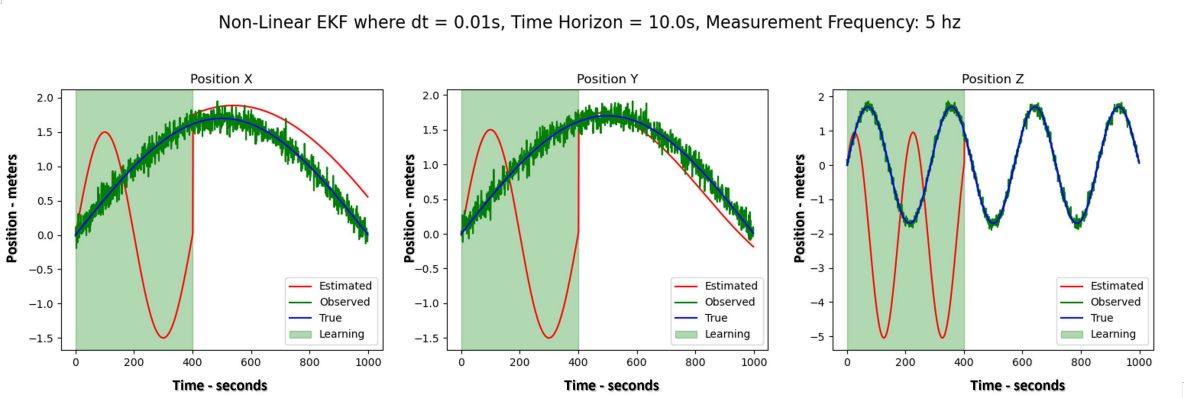


Figure 6.18: EKF without update and with learning - offline testing

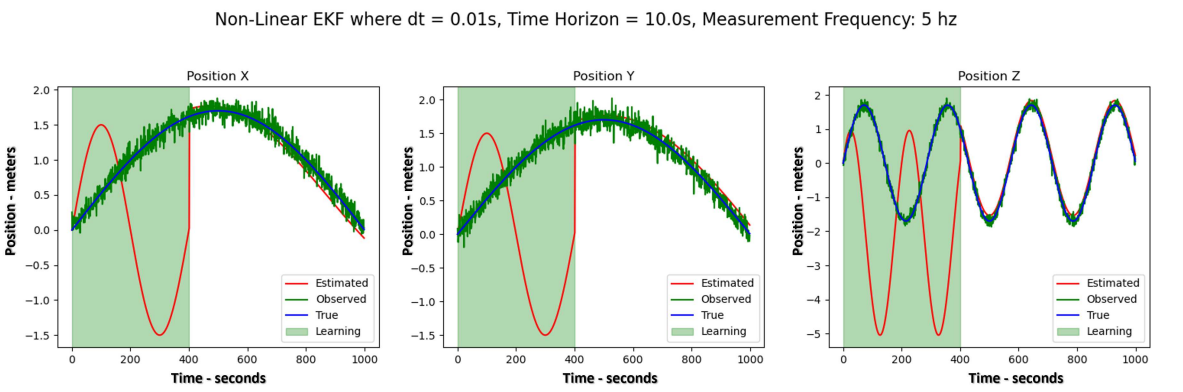


Figure 6.19: EKF with update and with learning - offline testing

All the results in 6.17 to 6.19 were conducted in an offline setting. To further test the proposed EKF, an online testing was conducted where I simulated the measurements by publishing their values on ros topics and then implement the EKF and run it online to estimate the garage's position. Figure 6.20 shows the online testing of the proposed EKF when the simulated measurements were following a sine pattern. After spending a few seconds observing and learning the parameters, the EKF was able to estimate the garage's position with good accuracy. To make it more complex and realistic, a sum of the sines pattern was used to simulate the measurements. The EKF was still able to learn the parameters and estimate the garage's position as illustrated in Figure 6.21

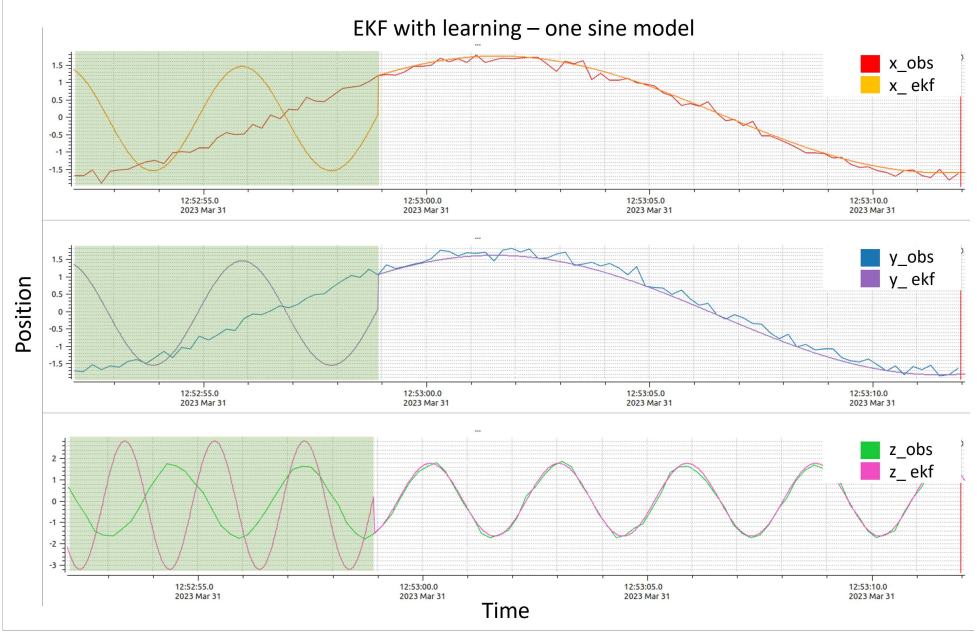


Figure 6.20: EKF with learning - one sine model - online testing

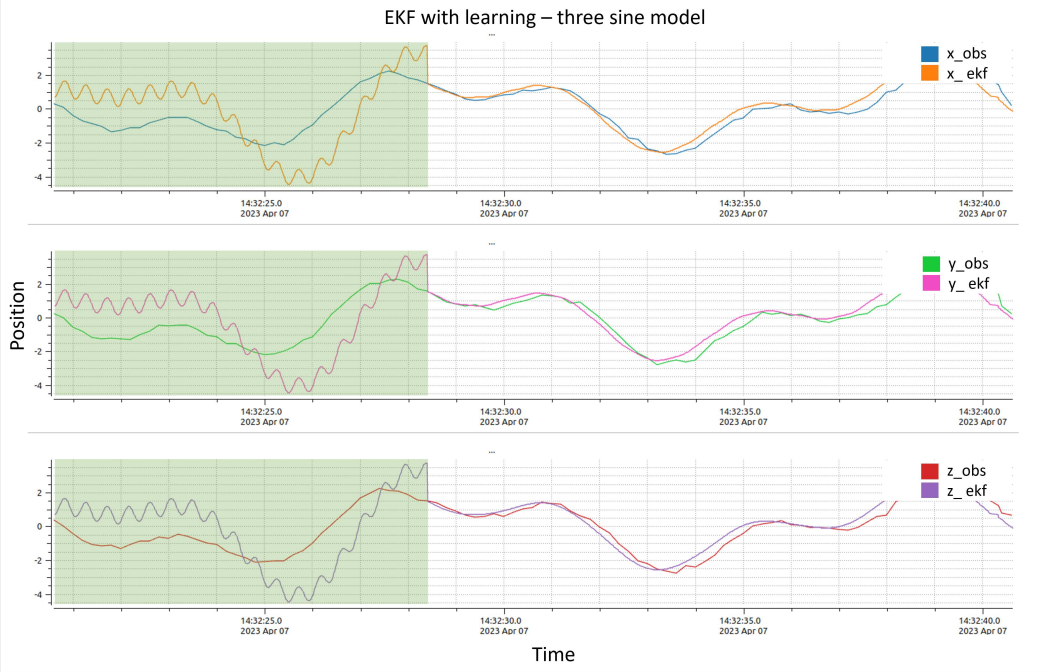


Figure 6.21: EKF with learning - three sine model - online testing

6.3 Garage detection

This section will discuss the results of the object detection algorithm using yolov5 as previously illustrated in section 5.2. At the beginning of this work, I did not have any access to real datasets so as a proof of concept, I trained the yolov5 on the garage model in the simulation. The results, of course, were great; the yolo was able to detect the garage from almost all angles and orientations. Around 300 images from the simulation were in training the yolo. Figure 6.22 shows an example of good detection which was the case most of the time. On the hand, there were some instances of bad detection when the garage is close to the camera as shown in Figure 6.23. This issue can be tackled by increasing the training dataset as well as making it more diverse.

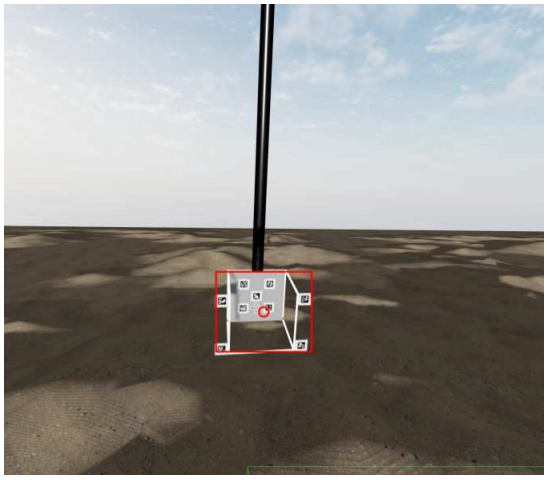


Figure 6.22: Garage detection using yolov5

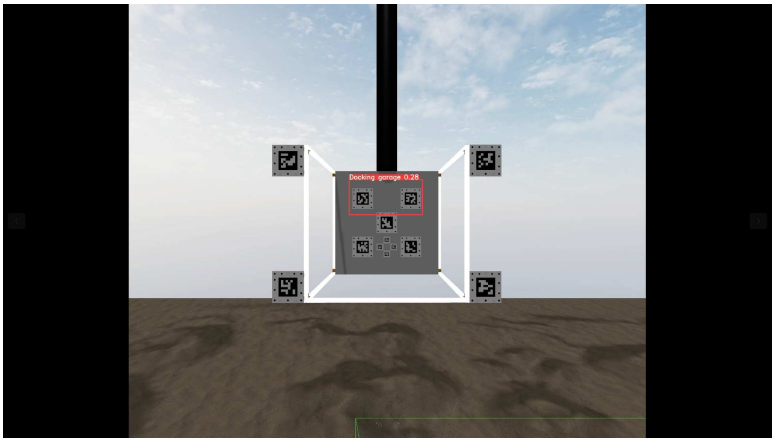


Figure 6.23: An example of bad detection

6.4 Collecting real dataset

This section will present some of the results from the collected dataset of the garage in Sète, South France. The main two goals of collecting this dataset were to record realistic garage behaviour and then later update the simulation accordingly. The second goal was to is to collect a training dataset for the object detection model. Figure 6.24 demonstrates the garage movement along the z-axis as well as the main orientation angles: pitch, roll, and yaw. The heave oscillations had a very small amplitude in the order of a few centimetres while the frequency was nearly 0.5hz. The main reason for that was because the sea was very calm at the time of the recording. Moreover both the pitch and yaw demonstrated robust and anticipated behaviour where the pitch should not change much and the yaw may change but in an incremental way. The roll, however, was aggressive and we believe they might be an issue with the sensor driver. Ideally, the roll should demonstrate a similar behaviour as the pitch.

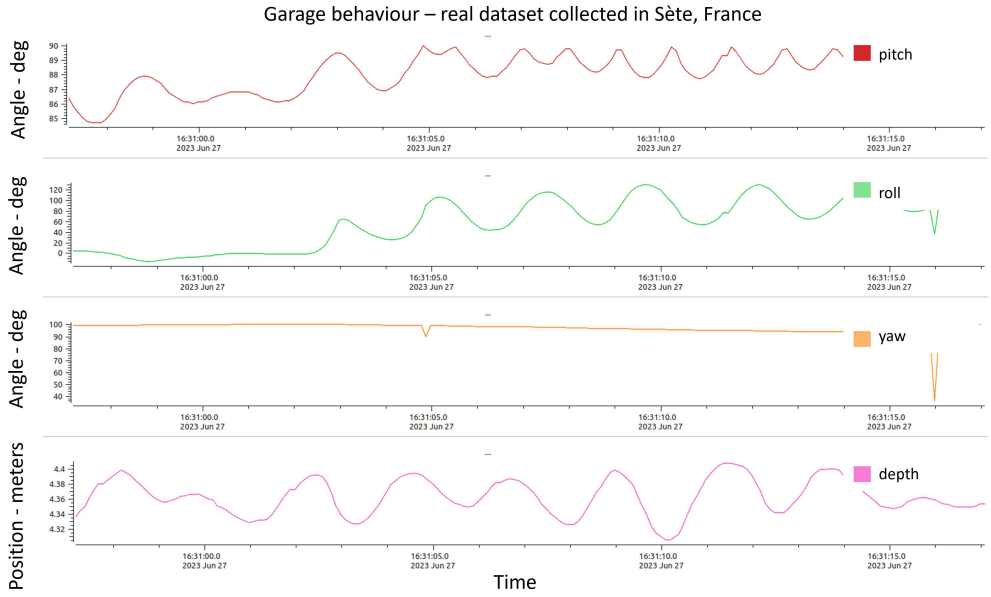


Figure 6.24: Garage behaviour - real trial conducted in Sète, France

Regarding collecting training for the yolo, the water visibility was very bad. We could not see the garage unless we are too close. Figures 6.25 and 6.26 show examples of the bad visibility that we encountered. Figure 6.27 shows an example of perfect visibility conditions during a previous trial in 2021.

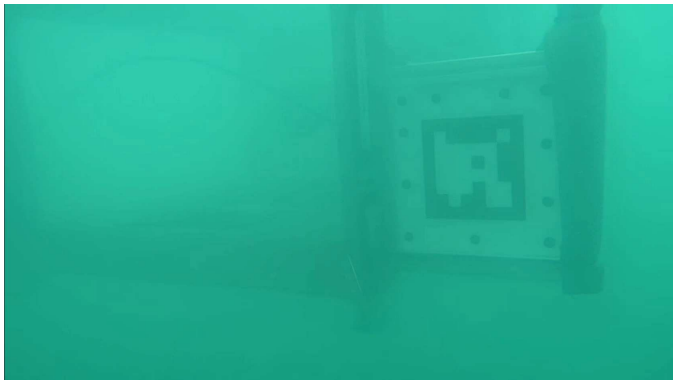


Figure 6.25: An example of poor visibility conditions at Sète

Figure 6.26: An example of poor visibility conditions at Sète

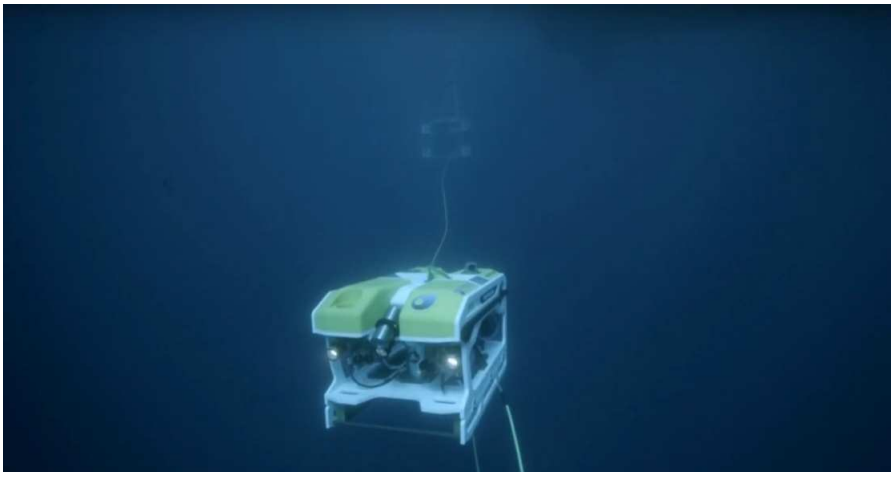


Figure 6.27: An example of ideal visibility conditions from a previous trial in 2021

CHAPTER 7

Conclusions and Future Work

People do not like to think. If one thinks, one must reach conclusions. Conclusions are not always pleasant - Helen Keller

Contents

7.1 Conclusions **67**
7.2 Future work **68**

In this chapter, the final outcomes of the project as well as the future improvements and recommendations will be discussed.

7.1 Conclusions

Many industries and offshore missions start to depend heavily on underwater robotics. One of the main challenges associated with using these vehicles is retrieving them after the mission. One of the proposed solutions is conducting autonomous docking since manual docking can be challenging even for experienced pilots as it depends on the sea state and environmental disturbances. There are several types of docking, stationary, non-stationary and towed docking. In this case study, I am focusing on non-stationary docking where the docking garage will be hung by a cable and subject to all kinds of disturbances. This work focuses mainly on developing solutions to improve the perception and decision-making capabilities in non-stationary autonomous docking. To do so, a custom simulation environment was developed in Gazebo to be able to simulate and test any future algorithm since testing on real hardware is expensive. The developed simulation included basically simulating the disturbances in the garage using different approaches. Another direction that was taken in this work was developing a motion es-

timation algorithm to estimate the garage's movement given that the markers detection may be interrupted due to visibility or oscillations. Moreover, a deep learning-based object detection approach was implemented to detect the garage in real time and use this information in the control strategy later. Last but not least, a control strategy was developed based on the outputs of the object detection algorithm to guide the robot toward the docking garage. Furthermore, a dataset from the real garage was collected in Sète, France last June with the goal of improving the implemented solutions. So this dataset included images of the garage from different orientations to help train the object detection model. It included also recordings of the garage heave oscillations and changes in orientation which will help in developing a realistic simulator.

7.2 Future work

For future work and recommendations, the motion estimation could be improved and tested on more realistic dataset. It can also be integrated with a control strategy to determine when to dock. In addition, collecting new datasets in much better visibility conditions would be encouraged in order to train the object detection model. Also collecting a dataset of the garage's behaviour at different sea states would be encouraged in order to implement several scenarios in the simulation as well as testing the motion estimation. Further research about using sonar to detect the garage and approach it in case of bad visibility would be a recommended research direction as well. Another recommendation would be to implement the control strategy as a behaviour tree as it will be much easier to debug and modify.

Bibliography

[1] FORSSEA ROBOTICS : Argos ROV during sea trials @ La Ciotat, 2020 — youtube.com. https://www.youtube.com/watch?v=71zSPeOX2uA. [Accessed 11-Jul-2023].

[2] Gazebo — staging.gazebosim.org. https://staging.gazebosim.org/docs/garden/architecture. [Accessed 18-Jul-2023].

[3] GitHub - ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite — github.com. https://github.com/ultralytics/yolov5.git. [Accessed 04-Jul-2023].

[4] Ignition Gazebo: Buoyancy Class Reference — gazebosim.org. https://gazebosim.org/api/gazebo/6.1/classignition_1_1gazebo_1_1systems_1_1Buoyancy.html. [Accessed 03-Jul-2023].

[5] Ignition Gazebo: Underwater vehicles — gazebosim.org. https://gazebosim.org/api/gazebo/5.0/underwater_vehicles.html. [Accessed 03-Jul-2023].

[6] ISD4000 - AUV ROV Depth Temperature Sensor with AHRS — impactsubsea.co.uk. https://www.impactsubsea.co.uk/isd4000/. [Accessed 17-Jul-2023].

[7] ROS 2 Documentation x2014; ROS 2 Documentation: Foxy documentation — docs.ros.org. https://docs.ros.org/en/foxy/index.html. [Accessed 11-Jul-2023].

[8] Rovins - iXblue — ixblue.com. https://www.ixblue.com/store/rovins/. [Accessed 11-Jul-2023].

[9] Spherical pendulum - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Spherical_pendulum. [Accessed 18-Jul-2023].

[10] Stability in Flotation — sciencedemonstrations.fas.harvard.edu. https://sciencedemonstrations.fas.harvard.edu/presentations/stability-flotation. [Accessed 03-Jul-2023].

[11] The works of Archimedes : Archimedes : Free Download, Borrow, and Streaming : Internet Archive — archive.org. https://archive.org/details/worksofarchimede00arch/page/256/mode/2up?view=theater. [Accessed 03-Jul-2023].

[12] Work Class ROV vision based autonomous docking.

[13] Work Class ROV vision based autonomous docking — youtube.com. https://www.youtube.com/watch?v=ypgwMfDN6rU. [Accessed 18-Jul-2023].

[14] Gazebo - underwater vehicles. https://gazebosim.org/api/gazebo/6.2/underwater_vehicles.html, 2017. Accessed: June 14, 2023.

[15] Kinder Chen. Denoising Data with Fast Fourier Transform — kinder-chen.medium.com. https://kinder-chen.medium.com/denoising-data-with-fast-fourier-transform-a81d9f38cc4c. [Accessed 18-Jul-2023].

[16] Veerle A.I. Huvenne, Katleen Robert, Leigh Marsh, Claudio Lo Iacono, Tim Le Bas, and Russell B. Wynn. *ROVs and AUVs*, pages 93–108. Springer International Publishing, Cham, 2018.

[17] Mainwaring, Steven R. asv_wave_sim - autonomous surface vehicle wave simulator. https://github.com/srmainwaring/asv_wave_sim, 2023. Accessed: June 14, 2023.

[18] Petar Trslic, Matija Rossi, Luke Robinson, Cathal W. O'Donnel, Anthony Weir, Joseph Coleman, James Riordan, Edin Omerdic, Gerard Dooly, and Daniel Toal. Vision based autonomous docking for work class rovs. *Ocean Engineering*, 196:106840, 2020.

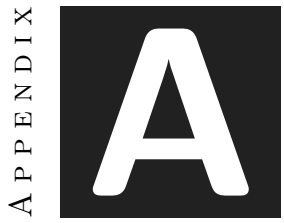
[19] Petar Trslic, Matija Rossi, Luke Robinson, Cathal W. O'Donnel, Anthony Weir, Joseph Coleman, James Riordan, Edin Omerdic, Gerard Dooly, and Daniel Toal. Vision based autonomous docking for work class rovs. *Ocean Engineering*, 196:106840, 2020.

[20] Petar Trslić, Edin Omerdic, Gerard Dooly, and Daniel Toal. Neuro-fuzzy dynamic position prediction for autonomous work-class rov docking. *Sensors*, 20(3):693, 2020.

[21] Petar Trslić, Edin Omerdic, Gerard Dooly, and Daniel Toal. Neuro-fuzzy dynamic position prediction for autonomous work-class rov docking. *Sensors*, 20(3), 2020.

[22] Yanxiang Wang, Honglun Wang, Bailing Liu, Yiheng Liu, Jianfa Wu, and Zhenyi Lu. A visual navigation framework for the aerial recovery of uavs. *IEEE Transactions on Instrumentation and Measurement*, 70:1–13, 2021.

[23] A.M. Yazdani, K. Sammut, O. Yakimenko, and A. Lammas. A survey of underwater docking guidance systems. *Robotics and Autonomous Systems*, 124:103382, 2020.



Other considerations

A.1 Appendices

Some supplementary videos demonstrating some aspects of this work are uploaded on the following link:

https://drive.google.com/drive/folders/1VADltiDcvNYImcWsRtYuvg_w8TyCQ_IC?usp=sharing

Moreover, all the papers that I have read during my state-of-the-art research are uploaded on the following link:

https://drive.google.com/drive/folders/13utF3S90LSrErx4A_hT40wCSbzW_rgyO?usp=sharing