UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

---

Study and development of NGC algorithms for a ROV system and its digital twin

Studio e implementazione di algoritmi NGC per un sistema ROV ed il suo gemello digitale

Candidate:
Flavia Gioiello

Advisor:
Prof. David Scaradozzi

Coadvisor:
Dott. Pierre Drap

Academic Year 2022-2023

What we know is a drop,
what we don't know is an ocean.
ISAAC NEWTON

Ringraziamenti

Prima di procedere con la trattazione di questa tesi, desidero prendere un momento per dedicare alcune parole a tutti coloro che mi hanno accompagnato in questo significativo percorso di crescita personale e professionale. Il viaggio che mi ha portato fino a questo punto non è stato soltanto un percorso accademico, ma anche un cammino ricco di esperienze umane e incontri che hanno lasciato un'impronta indelebile nella mia vita.

In primis, un ringraziamento speciale al mio relatore di tesi, il professore David Scaradozzi, per la sua guida inestimabile, il suo sostegno e la sua esperienza. Grazie per aver riposto in me una grande fiducia e per aver fatto una scommessa sul mio potenziale. Sono grata per avermi introdotto nel vasto e affascinante mondo della robotica marina, ambito in cui spero di poter proseguire il mio percorso in futuro.
Desidero esprimere la mia sincera gratitudine anche al mio correlatore, Pierre Drap e a Mohamad Motassem Nawaf, per la loro guida costante e per avermi trattato alla pari nel corso di questo progetto. La loro fiducia e l'opportunità che mi hanno dato sono state fondamentali nel mio sviluppo professionale e personale. Sono onorata per aver lavorato al vostro fianco.

Ringrazio infinitamente i miei genitori e la mia famiglia, per il loro amore incondizionato, il sostegno e la pazienza. Hanno fornito un rifugio di pace e motivazione nei momenti più impegnativi e non hanno mai esitato nel credere nel mio valore, supportandomi sempre con grande forza.

Grazie ad Alessia, che da 8 anni con immensa pazienza ascolta le mie lamentele e mi dà conforto e sostegno anche a distanza, a qualsiasi ora del giorno e della notte.
Grazie a Federica, Hermes e Regy, che mi hanno accompagnato e sostenuto lungo tutto il mio percorso universitario, regalandomi sempre un sorriso a fine giornata e condividendo con genuino entusiasmo i miei successi.
Grazie ad Alessandro, Michele, Mauro e Simone, per la vostra accoglienza calorosa e i momenti di spensieratezza che abbiamo condiviso, i quali hanno illuminato anche i periodi più impegnativi di questo percorso universitario.
Grazie ai miei compagni di progetto, Lorenzo, Adelio, Fiore e Davide, con i quali ho condiviso sfide impegnative e successi, superando ostacoli e momenti di sconforto.
Un grazie speciale va a Martina, mia compagna di viaggio e grande guida durante il tirocinio. Grazie per aver ricoperto il ruolo di "sorella maggiore" e per aver condiviso

momenti indimenticabili con me.
Ringrazio con il cuore tutte le persone conosciute durante il periodo all'estero, per avermi fin da subito accolta con affetto: con ognuno ho instaurato dei profondi rapporti di amicizia. Grazie al vostro sostegno avete reso questa esperienza cruciale per la mia crescita personale.

Infine, ma non per importanza, dedico l'intera tesi a mia nonna Giulia, da sempre fonte di grande forza e pilastro portante della famiglia e che da lassù immagino mi guardi infinitamente orgogliosa di me ed esclami con ironia: "Sei la vergogna della famiglia!".

Ancona, Novembre 2023

Flavia Gioiello

Abstract

This thesis highlights the critical role of submarine vehicles, with a particular emphasis on the deployment of a Remotely Operated Vehicle (ROV), in enhancing the exploration and understanding of the underwater domain. The primary purpose is to devise a guidance, navigation and control system for a BlueROV2, a product of BlueRobotics, integrated with the SeaTrac Lightweight USBL navigation system by Blueprint Subsea. The guidance system is designed to compute instructions for the ROV's thrusters, facilitating autonomous control over its trajectory and orientation. At the core of the navigation system is a complex set of sensors, including compasses, USBL positioning systems, mono and stereo optical systems, Inertial Measurement Units (IMUs), altimeters, depth meters, and other relevant technologies. A Line of Sight control strategy was developed to mantain a linear trajectory, while a digital twin of the ROV, in Unity, mirrors its behavior in real time. This dual approach synergistically harnesses the physical attributes of the ROV along with the advanced simulation and analytical capabilities of its digital analogue, providing a comprehensive methodology for detailed seabed mapping and image processing. The study concludes with the exhibition of experimental outcomes acquired during a maritime expedition in the Port of Marseille, demonstrating the efficacy and efficiency of the implemented control system. The entire work was realized during the internship for the ERASMUS program at the CNRS (Centre national de la recherche scientifique) at the University of Aix-Marseille.

Sommario

Questa tesi esamina l'importanza fondamentale dei veicoli sottomarini, in particolare di un Remotely Operated Vehicle (ROV), nel promuovere l'esplorazione e la comprensione degli ambienti marini. L'obiettivo primario è lo sviluppo di un sistema di guida, navigazione e controllo per un BlueROV2, un prodotto di BlueRobotics, integrato con il sistema di navigazione SeaTrac Lightweight USBL di Blueprint Subsea. Il sistema di guida è progettato per generare istruzioni per i propulsori del ROV, facilitando il controllo autonomo sulla sua traiettoria e il suo orientamento. Al centro del sistema di navigazione vi è un complesso insieme di sensori, inclusi bussole, sistemi di posizionamento USBL, sistemi ottici mono e stereo, Unità di Misura Inerziali (IMU), altimetri, misuratori di profondità e altre tecnologie pertinenti. È stata sviluppata una strategia di controllo Line of Sight per mantenere una traiettoria lineare, mentre un gemello digitale del ROV, in Unity, ne replica il comportamento in tempo reale. Questo approccio duale sfrutta sinergicamente gli attributi fisici del ROV insieme alle avanzate capacità di simulazione e analisi del suo analogo digitale, fornendo una metodologia completa per la mappatura dettagliata dei fondali marini e l'elaborazione delle immagini. Lo studio si conclude con la presentazione dei risultati sperimentali acquisiti durante una missione marittima nel Porto di Marsiglia, dimostrando l'efficacia e l'efficienza del sistema di controllo implementato. L'intero lavoro è stato realizzato durante il programma ERASMUS presso il CNRS (Centre national de la recherche scientifique), all'Università di Aix-Marseille.

Contents

Introduction 1

1 Background 3
- 1.1 Remotely Operated Vehicles . . . 3
- 1.2 Navigation systems and sensors . . . 7
  - 1.2.1 Inertial sensors . . . 8
  - 1.2.2 Acoustic sensors . . . 10
  - 1.2.3 Geophysical sensors . . . 12
- 1.3 Virtual simulation of marine environments . . . 14
  - 1.3.1 Photogrammetry . . . 16
  - 1.3.2 Digital Twin . . . 19

2 Mathematical model for design and control 21
- 2.1 Kinematic model . . . 22
- 2.2 Hydrodynamic model . . . 25
  - 2.2.1 Inertial matrix and added mass . . . 26
  - 2.2.2 Centripetal and Coriolis matrix . . . 27
  - 2.2.3 Hydrodynamic damping matrix . . . 28
  - 2.2.4 Vector of gravitational and buoyancy force . . . 29
  - 2.2.5 Thrusters allocation matrix and forces . . . 29
- 2.3 Parameters of a BlueROV2 . . . 31
- 2.4 State-space model . . . 31

3 Approach 33
- 3.1 Materials and Methods . . . 33
  - 3.1.1 Underwater Setup . . . 33
  - 3.1.2 Surface Setup . . . 39
  - 3.1.3 Software Applications . . . 42
  - 3.1.4 Final Architecture . . . 44
- 3.2 Development of a LOS control system . . . 45
  - 3.2.1 Line of Sight Guidance . . . 45
  - 3.2.2 Implementation of the NGC system . . . 46
- 3.3 Development of an experimental Digital Twin environment . . . 51
  - 3.3.1 Unity Digital Twin . . . 52
  - 3.3.2 Implementation of the NGC system . . . 53

4 Experimental results **59**
4.1 Tests of the NGC system 59
4.2 Simulation-Based Performance Analysis 64

Conclusions **71**

List of Figures

1.1 Components of a ROV. (Image credit: Saab Seaeye) [1] 5
1.2 Classification of navigation systems and their respective sensors. (Image credit: AUV Navigation and Localization: A Review [2]) 8
1.3 Acoustic position systems: (a) SBL, (b) USBL, (c) LBL 12
1.4 How colors penetrate in different ways in oceans. (Image credit: University of Minnesota Sea Grant Program) 15

2.1 Earth-fixed and Body-fixed frames 22

3.1 BlueROV2 with Heavy Configuration Retrofit Kit: (a) top view, (b) front view. 34
3.2 Raspberry Pi 4 36
3.3 Navigator Flight Controller. (Image credit: BlueRobotics) 36
3.4 Fanthom ROV Tether. (Image credit: BlueRobotics) 37
3.5 X150 Beacon 39
3.6 GPS receiver BU-353S4 40
3.7 Switcher for ROV's communication 40
3.8 Outland Technology Power Supply 41
3.9 ArduSub logo. (Image credit: BlueRobotics) 42
3.10 MAVLink logo. (Image credit: mavlink.io) 42
3.11 QGC logo. (Image credit: qgroundcontrol.com) 43
3.12 Ros Noetic logo. (Image credit: ROS.org) 44
3.13 Unity logo. (Image credit: unity.com) 44
3.14 Heading LOS guidance strategy 46
3.15 Complete configuration 47
3.16 QGC local port configuration 48
3.17 QGC auto-connection configuration 48
3.18 Flowchart diagram of the software 51
3.19 Digital representation of the dock in Unity 52
3.20 Digital representation of the seabed in Unity 53
3.21 Digital Twin of the ROV 53
3.22 Complete Configuration 54
3.23 ROV-Python-ROS-Unity communication 54
3.24 Flowchart diagram of the second software 57

4.1 ROV's Buoyancy 60

4.2 Desired latitude and longitude and GPS position 60
4.3 3D trajectory graph (lat-lon-depth) 61
4.4 Time - Depth graph 62
4.5 Roll, Pitch, Yaw graphs 63
4.6 Time - Lat,Lon Graph 63
4.7 Time - Speed Graph 64
4.8 ArduSub Simulator: (a) parameters setting (b) terminal 66
4.9 MAVProxy Console 67
4.10 ROS terminal 68
4.11 Unity-ROS connection 68
4.12 QGC localization map 68
4.13 Rov's path 69
4.14 Python script terminal: (a) initialization, (b) new waypoints scanning 70

List of Tables

1.1 A comparison between ROVs and human divers for underwater explorations. 4

2.1 Notation used in underwater vehicle dynamics, detailing movements, positions, velocities, and forces. 23
2.2 Moment arms of the 8 thrusters of the BlueROV2. 31
2.3 BlueROV2 Parameters 32

Acronyms

AHRS Attitude and Heading Reference System. 9, 38

AUV Autonomous Underwater Vehicle. 1, 11

BCC Brightness Constancy Constraint. 16

DOF Degrees Of Freedom. 9, 18, 21, 22, 25, 29, 34, 38

DVL Doppler Velocity Log. 9

GPS Global Positioning System. 10

IMU Inertial Measurment Unit. 9, 10, 35

LBL Long Baseline. 11

LOS Line of Sight. 45, 61

LUSBL Long & Ultra Short Baseline. 11

NED North-East-Down coordinate system. 22, 49

PWM Pulse Width Modulation. 34

ROS Robot Operating System. 52

ROV Remotely Operated underwater Vehicle. 1, 3, 21, 33

SBL Short Baseline. 11

SfM Structure from Motion. 19

SIFT Scale-Invariant Feature Transform. 17

SLAM Simultaneous Localization and Mapping. 7, 19

SNAME Society of Naval Architects and Marine Engineers. 23, 27

SONAR SOund Navigation And Ranging. 12

TOF Time Of Flight. 8, 11

USBL Ultra-Short Baseline. 11

UUV Unmanned Underwater Vehicle. 1, 9

Introduction

Covering over 70% of the Earth's surface, oceans constitute the planet's largest ecosystem, holding 99% of all habitable space. Despite that, the depths of the underwater realm remains predominantly uncharted, as only 5% of the seabed has been explored, containing secrets crucial to understand the planet's history, climate, and biodiversity.[3]
Thus, oceanic exploration emerges as one of the most captivating frontiers in the quest for scientific knowledge and technological advancement.
However, the high pressure, low temperatures, and the darkness characteristic of the deep sea present significant obstacles to scientists, especially for the acquisition of high-quality underwater images for photogrammetry. Moreover, conducting researches in these conditions necessitates specialized certification, stringent safety protocols, and robust physical endurance. The combination of these factors makes underwater surveys not only technically demanding but also a real hazard.[4]
It is evident that submarine vehicles play a crucial role by offering versatility, precision, and safety features to explore environments that would otherwise be inaccessible. Underwater vehicles can be categorized into two distinct types: "manned" and "unmanned" vehicles. Manned vehicles are essential for missions that require direct human involvement, offering the advantage of immediate data interpretation. In contrast, an Unmanned Underwater Vehicle (UUV) represents a safer solution to the challenges posed by the hostile environment.[5] These vehicles include an Autonomous Underwater Vehicle (AUV), which is programmed to navigate autonomously without direct human guidance, and a Remotely Operated underwater Vehicle (ROV), which is controlled remotely by human operators. ROVs can also acquire the benefits of an AUV by implementing special guidance and control strategies. Therefore, ROVs prove to be ideal for acquiring underwater images as they can adjust their velocity and distance from a target according to requirements and protecting the environment. However, using a ROV presents some challenges, like the vehicle's autonomy. ROVs could certainly go to great depths during their surveys, requiring to be operational for an extended period. It is, also, required a good bouyancy and balance, while scanning the seabed. That is not always guaranteed, due to strong sea currents.

Especially, the objective of this master thesis is to present a possible application for underwater photogrammetry using a ROV, giving possible solutions to the challenges shown above and illustrating algorithms to make the robot autonomous in following a predetermined path up to specific coordinates.

Acronyms

The research conducted as part of this study was advanced within the scope of underwater robotics at LabMACS - DII - UNIVPM. It builds upon and expands the work presented in Martina Rossi's thesis[6], testing it in a real-world environment. A high-fidelity digital-twin of the vehicle was designed, offering a virtual environment for testing and training. A digital-twin can be used to simulate challenging scenarios, refine control algorithms, and predict equipment failures, all while the actual ROV operates in the deep sea.

This thesis is organized as follow:
Chapter 1 presents a brief overview on Remotely Operated Vehicles, their features and applications and related researches in the area of navigation systems. It also provides an examination of photogrammetry techniques for the purpose of replicating the marine environment with the digital model.
Chapter 2 analyses the mathematical model behind the ROV's locomotion, especially reference systems, the rotational matrix, kinematic and dynamic models, parameters of the BlueROV2 and the space-state model.
Chapter 3 describes the materials and software methods used, as well as the communication between them. Furthermore, it analyzes the logic underpinning the control system, illustrating solutions implemented for two applications: a localization and control challenge in reaching a specific latitude and longitude point, and an experimental fusion of the real ROV with its digital twin for scanning purposes.
The collected results of the first application are shown in **Chapter 4**. The second application was not tested in a marine environment due to a malfunction in the ROV.
Finally, **Conclusions** summarizes the key findings of the study and their implications, interpreting the results, underlining the issues in this navigational system and introducing some considerations about future developments.

Chapter 1

Background

This chapter is about the state of the art of Remotely Operated Vehicles (ROVs) and their navigation and control systems. ROVs have evolved significantly over the years, being used in a variety of underwater applications. The chapter begins by providing an overview of the historical development of ROVs, focusing on their features and characteristics.
An overview about sensors used in submarine vehicles and the latest navigation systems is also presented, highlighting how these systems facilitate precise maneuvering and stability in unpredictable marine conditions. The chapter also provides an examination of the concept of digital twins and photogrammetry techniques and why they are particularly powerful tools in this context.

1.1 Remotely Operated Vehicles

A Remotely Operated underwater Vehicle (ROV), often referred to as an underwater drone, is controlled by an operator placed on the surface, who can navigate it staying safe.
The utility of ROVs becomes particularly evident when considering the limitations of human divers and manned submersibles. While SCUBA divers are generally restricted to depths of no more than one hundred meters due to safety and physiological constraints, ROVs can operate at much greater depths. [1] In addition to this, ROVs are a low-cost solution to explore seabeds, not requiring time-consuming diving training.
The table 1.1 shows a comparison between the two different ocean exploration methods, diving or using a ROV. ROVs are equipped with tethered cables, enabling real-time data transmission from the ocean while divers can be equipped with cameras which are later processed. Regarding the supply, ROVs can operate continuously in marine environments as long as their batteries are regularly replaced, while the divers' equipment impact their mobility and endurance. Divers, also, are exposed to risks such as decompression sickness, nitrogen narcosis, and equipment issues, requiring training to handle emergency situations. [7]
Therefore, due to these advantages over human divers, ROVs have become an essential component in the field of marine exploration and research.

Characteristics	**Divers**	**ROV**
Data Transmission	Usually not supported	A theter cable that connects the ROV to its support ship
Dimensions	Bulky due to diving equipment	Small-sized and flexibility to reach hostile places
Movements	Depends on swimming abilities	Limitations due to the cable and thrusters
Power supply	Limited by the air tanks and physical endurance	Batteries or with cables
Safety	Unsafe	Only risk of malfunctions

Table 1.1: A comparison between ROVs and human divers for underwater explorations.

The history of ROVs dates back to 1953 when the French pioneer Dimitri Rebikoff created the "Poodle", an underwater scooter equipped with a tether to allow it to be controlled from the surface by a pilot.
In the 1960s, the United States Navy began using ROVs for recovery missions and simple tasks. An example is the able-Controlled Undersea Recovery Vehicle (CURV), developed to recover a hydrogen bomb in the Palomares incident in 1966.
During the 1970s the oil industry recognize the potential of these vehicles for offshore operations. This period saw the evolution of ROVs from basic tethered devices to more sophisticated systems with greater maneuverability and video transmission.
ROVs continued to expand, reaching over 500 units in the 1980s, used also for commercial purpose.
Nowadays ROVs are equipped with high-definition cameras, advanced sensors, and robotic arms, and are used in a wide range of applications, from scientific researches and environmental monitoring to underwater construction and military operations.[1]

Despite ROVs are designed in different shapes and sizes based on specific requirements, they typically share several core components, shown in figure 1.1:

- **Thrusters**: ROVs are equipped with multiple thrusters, that are electrically or hydraulically powered propellers, placed on different parts to provide movements in all directions.
- **Tether**: A cable that connects the ROV to its controlling station for the power supply and data transmission, including live video feeds, sensor data, and operational instructions. In advanced ROVs, the tether might contain fiber-optic cables, which allow for high-speed data transmission. This cable is used for communication because radio waves attenuate rapidly in water,

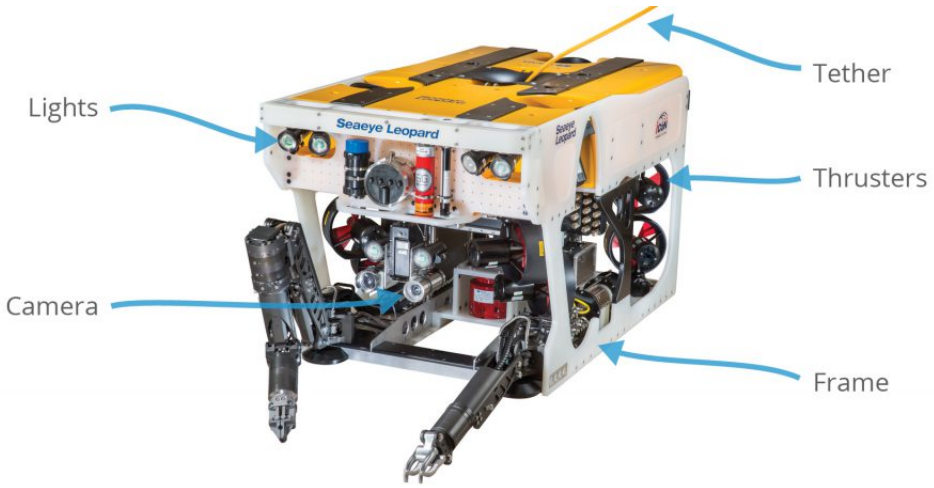


Figure 1.1: Components of a ROV. (Image credit: Saab Seaeye)[1]

making long-distance wireless communication impractical.

- **Lights**: The lighting systems is indispensable for visibility since the darkness of the underwater environment.
- **Camera**: The onboard camera serves as the only visual for the operator to navigate and guide the vehicle, providing images with low-latency.
- **Frame**: The frame provides the structural integrity necessary to withstand underwater pressure and protect internal components. It consists of a rigid, open frame, generally rectangular in shape, on which essential components such as thrusters, cameras, lights, the tether, and other sensors are mounted. This frame can be custom-designed to minimize water resistance and improve the hydrodynamic efficiency, impacting the ROV's performances.
- **Pilot Controls**: The surface control station can vary in complexity, from a sophisticated control room to a simple smartphone or joystick connected to a monitor. Regardless of its form, the primary function is to facilitate communication and to display data and feedback, including the vehicle's location, allowing operators to remotely guide the ROV, monitor its environment through cameras and sensors, and make real-time decisions based on the information received.

In addition to these compenents, sensors, robotic arms, sonars, and other specialized equipment can be installed on the vehicle, making the ROV incredibly versatile for a lot of tasks.
For this reason, ROVs can be classified into 5 groups, based on size and purpose, as in [8]:

- **Class I - Observation ROVs**: These ROVs are characterized by their small size, maneuverability, and the specific functionalities for inspection tasks in

marine environments. They are equipped with essential components such as high-quality cameras, lighting, and basic navigational systems. They are tipically powered by on-board batteries or a tether, weigh up to 40 Kg and they can reach depths in the order of hundreds of meters.

- **Class II — Observation ROVs with Payload Option**: Unlike the first class, this type of ROVs can be equipped with more sensors and manipulators. They can have two simultaneously viewable cameras and a weight up to 300 Kg.

- **Class III — Work-class ROVs**: Class III ROVs have a higher payload capacity, allowing them to carry more equipment, including multiple manipulator arms, advanced sensors, and heavier tools, with a weight more than 300 Kg. They are powered by cables from the surface due to their high energy consumption, allowing them to reach several thousand meters of depth.

- **Class IV — Towed and bottom-crawling ROVs or Excavators**: These ROVs are designed to move along the seabed, equipped with tracks or wheels. They are essential for tasks like drilling, sediment sampling, or archaeological studies, that require stability, large area coverage or direct interaction with the seabed. They have reduced maneuverability and agility due to the size, with some models weighing as much as 5000 kg.

- **Class V — Prototype**: They are developmental and flexible vehicles used for testing new technologies or for research purposes.

Furthermore, ROVs can be categorized based on their weight, as in [9]:

- **Micro ROVs**: These are the smallest and lightest, often weighing less than 5 kg. They are typically used for simple inspection tasks in confined spaces.

- **Mini ROVs**: Weighing between 5 to 32 kg, mini ROVs offer greater capabilities, including more advanced sensors and cameras, while still being relatively easy to deploy and operate.

- **Large ROVs**: These ROVs weigh between 32 to 90 kg and are more robust, capable of carrying out a wider range of tasks including more complex inspections.

- **Shallow ROVs**: They are low-power vehicles with either copper or fiber-optic telemetry systems with a depth capability of up to 1000 meters.

- **Deepwater ROVs**: Similar to the shallow ROVs, they are designed for operations in deep marine environments, equipped with single or dual light manipulator systems and an high-voltage power.

- **Heavy ROVs**: They are designed for more demanding underwater tasks, equipped with electric thrusters, dual medium-duty hydraulic manipulators and a hydraulic power unit.
- **Standard work class ROVs**: These ROVs are typically in the 100 to 200 horsepower range, used for drill support in the offshore oil and gas industry or light construction work.
- **Heavy work class ROVs**: Characterized by their large size and power, typically exceeding 200 horsepower, these ROVs are designed for heavy construction work.

1.2 Navigation systems and sensors

An underwater vehicle must overcome three challenges to move autonomously, addressing the so-called problem of navigation.
The first one is the problem of **localization**; the robot must accurately estimate its position, either from a known starting point or within a pre-existing or dynamically constructed map, which can evolve based on the robot's movements. This process may involve complex algorithms and sensors, such as Simultaneous Localization and Mapping (SLAM) technologies.
The second challenge is **target identification**; the robot must determine its destination from a starting point and identify any intermediate waypoints. This involves not only spatial awareness but also the capability to make decisions based on its objectives and environmental data, which may include AI and machine learning algorithms for adaptive navigation.
The third problem is the **trajectory planning**, which requires the robot to establish an optimal path between its starting point and destination, adhering to specific parameters such as obstacle avoidance and efficiency. This necessitates advanced computational techniques, like path planning algorithms and real-time data processing.[10]

To overcome these challenges, a combination of different navigation systems can be utilized to achieve both the robustness and reliability of the overall navigation solution. Generally, these systems can be divided into three main categories (a schema is shown in figure 1.2) [2]:

- **Inertial Navigation System**: These systems use gyroscopes and accelerometers to calculate the vehicle's position, direction, and speed, independent of external references. INS is necessary in environments where external signals are unavailable, providing continuous and accurate navigation data. However, over time, the errors in position and velocity estimates can accumulate.
- **Acoustic Navigation System**: Utilizing sonar technology, acoustic systems can determine the vehicle's position relative to reference points by emitting

sound waves and measuring their Time Of Flight (TOF). This method is crucial for obstacle avoidance and for tasks that require precise positioning. Even so, they are influenced by factors like water temperature, salinity, and sound speed.

- **Geophysical Navigation System**: These systems use the Earth's magnetic field, gravitational field, or the topography of the ocean floor as reference points for navigation. They include also optical sensors and sonars. Geophysical navigation is particularly useful in long-range, deep-sea missions where other forms of navigation may be less effective. On the other hand, the environmental variables can distort the signals used for navigation, leading to inaccuracies.

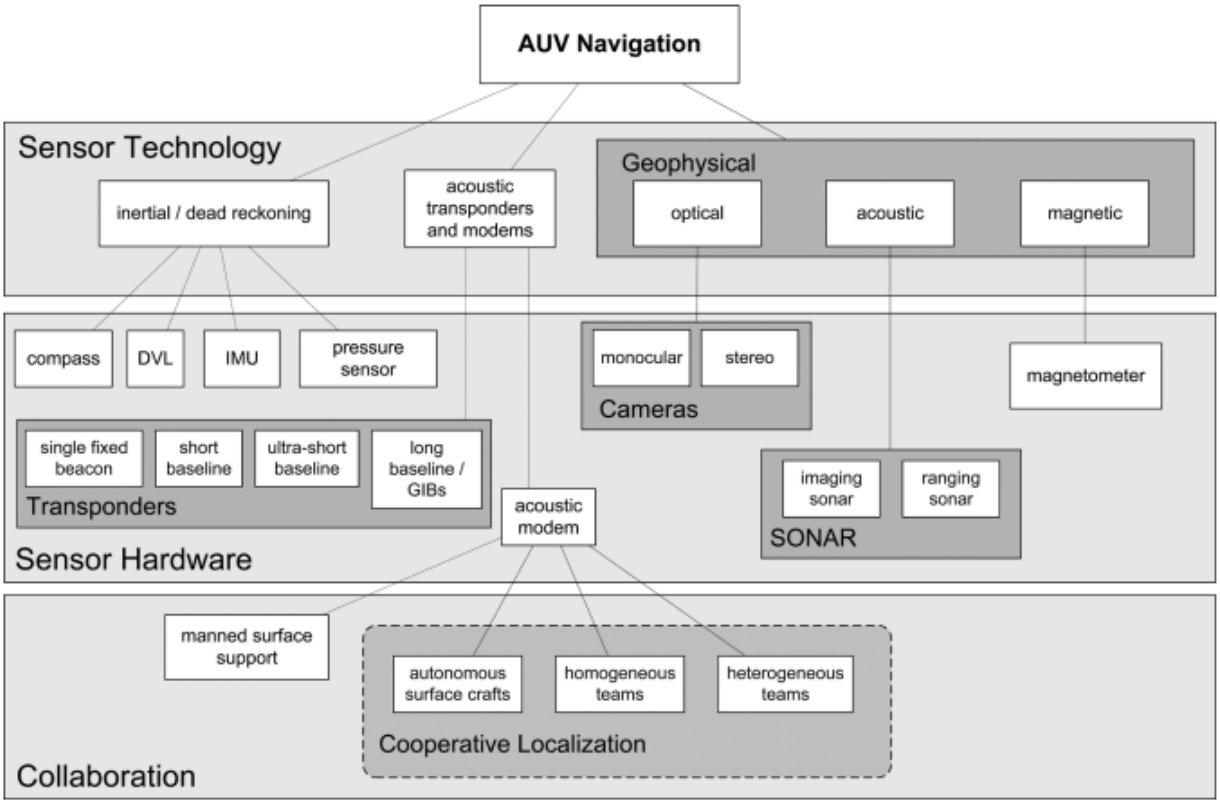


Figure 1.2: Classification of navigation systems and their respective sensors. (Image credit: AUV Navigation and Localization: A Review [2])

The following subsections offer an examination of the sensors used by each navigation system, highlighting their operational mechanisms and features.

1.2.1 Inertial sensors

The basic ideas behind inertial navigation are based on the Newton's laws of motion. A Inertial Navigation System consists of a combination of accelerometers, gyroscopes, and sometimes magnetometers. Accelerometers measure linear acceleration, while

gyroscopes measure angular velocity in order to calculate changes in velocity and orientation over time, a process known as "*dead reckoning*".
The most commonly installed inertial sensors in UUV include:

- **Inertial Measurment Unit (IMU)**: It is an electronic sensor, composed by accelerometers (mechanical, capacitive, piezoelectric, piezoresistive or heated gas) and gyroscopes (Mechanical, Laser, Fiber optic). Typically, these sensors are arranged in a "*strapdown*" configuration, where the sensitive axes of the inertial sensors are aligned orthogonally within a Cartesian coordinate system. An alternative configuration is the "*skewed*" where the gyroscopes and accelerometers are positioned on a cone that is angled relative to the vehicle, covering against faults, due to the redundancy of components. Compared to the strapdown configuration, the skewed one introduces more errors. As a result, it demands higher-quality gyroscopes and precise knowledge of the tilt angle to avoid significant misalignment errors. In modern IMU sensors, a magnetometer has been introduced to measure the three dimensions Earth's magnetic field, improving from a 6 DOF to a 9 DOF configuration.[2]

- **Doppler Velocity Log (DVL)**: The DVL is an acoustic sensors tipically used in Inertial Navigation Systems, since it provides essential velocity measurements for dead reckoning. Operating on the Doppler effect principle, the DVL emits acoustic pulses towards the seabed and measures the frequency shift in the reflected sound waves to estimate the relative vehicle's velocity. This sensor estimates the three-dimensional velocity measurements, providing data for forward/backward, lateral, and vertical movements. They require precise calibration and can be affected by varying of water conditions.

- **Pressure Sensor**: These sensors measure the pressure exerted by the surrounding water, to calculate depth. They can be sensitive to temperature changes, affecting their readings.

- **Compass**: A compass provides a globally referenced directional bearing. The most common type is the magnetic compass, which determines the vehicle's orientation relative to the Earth's magnetic field. Its accuracy can be compromised in the presence of objects with a strong magnetic signature, as it aligns with the Earth's magnetic north pole, which is different from the true geographic north. Another type is the gyrocompass, which uses a fast-spinning rotor and the principles of gyroscopic inertia and precession to align with true North, independent of the Earth's magnetic field. It finds true North by aligning with the Earth's rotational axis. So they are not affected by magnetic anomalies but require power to operate.

- **Attitude and Heading Reference System (AHRS)**: It is an advanced sensor system to provide accurate orientation information about the vehicle.

It combines sensors like accelerometers, gyroscopes, and magnetometers, to calculate the attitude (pitch, roll, and yaw). It is distinct from a IMU because it includes a microprocessor for the resolution of the alignment equations.

In many advanced Inertial Navigation Systems, GPS and IMUs are often used together.
GPS can provide absolute vehicle positioning but its effectiveness is affected underwater due to radio signal attenuation. So in integrated navigation systems GPS, from the surface, provides position information, while the IMU offers continuous data on orientation and movement, compensating for any short interruptions in GPS signal. However, the vehicle must periodically come to the surface to calibrate the position. In combination with these devices, an Extended Kalman Filter is usaually used to improve position estimation while the GPS signal is unavailable.[11]
The concept of emulating GPS technology in underwater environments can be categorized into three distinct groups[12]:

- ***"False"* underwater GPS**: An underwater vehicle drag a surface buoy equipped with a GPS. The GPS device communicate with the vehicle through a cable or fiber. This method, however, doesn't provide the exact location of the target but rather a proximate position, often within a few tens of meters from the surface buoy; that's why it is called *"False"* GPS.
- ***"Direct"* underwater GPS**: In this system, surface buoys equipped with GPS transmit acoustic waves directly to underwater receivers, mounted on the underwater vehicles. Unlike traditional GPS, which transmits radio-electric signals, this method uses acoustic signals. The underwater vehicle receives these acoustic signals and then computesw its own position.
- ***"Reverse"* underwater GPS**: This type is similar to the direct one but operates on a reverse principle. The underwater vehicle is equipped with an acoustic emitter and the surface buoys are equipped with submerged hydrophones and GPS receivers. These buoys measure the time of arrival of acoustic signals emitted from the underwater target, estimating its position.

These methods can be applied not only for inertial approaches but also for acoustic ones.

1.2.2 Acoustic sensors

Classic electromagnetic based communication techniques for flying robots, become ineffective for underwater applications, particularly at greater depths, due to the absorption of electromagnetic waves by water. In contrast, as a denser medium than air, water facilitates better transmission of sound waves.
As a result, acoustic navigation systems have gained popularity for facilitating communication in underwater environments. They operate by emitting sound pulses and

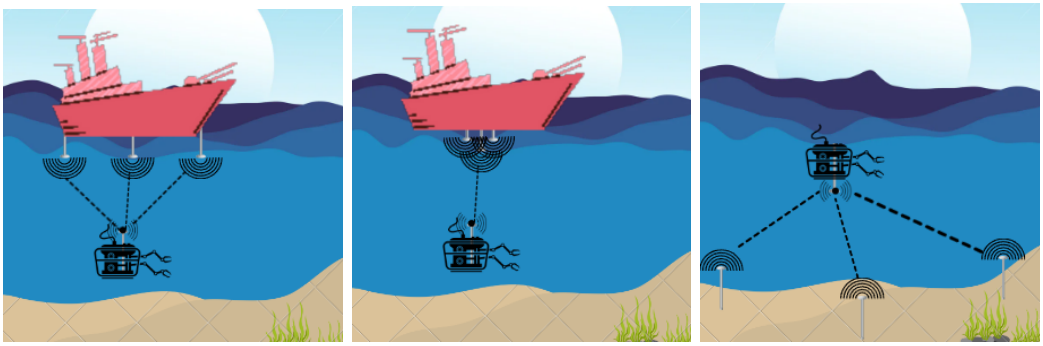
measuring their reflections from the seafloor or other structures. The time it takes for the sound waves to return (TOF) is used to calculate distances.[12]
However, Acoustic Navigation Systems involves certain challenges and issues, as a limited bandwidth, necessitating the use of time-division multiple-access (TDMA) techniques for information sharing among nodes. Other problems are a low data transmission rates, high latency, as sound travels through water at a speed of 1500 m/s, much slower than light; this speed can be influenced by changes in water temperature and salinity, affecting signal consistency. Therefore, these systems are used in calm weather conditions and in the absence of boat movement, in opposition to real conditions.

It is possible to differentiate between various configurations for acoustic communication, shown in figure 1.3:

- **Short Baseline (SBL)**: A set of three hydrophones, are installed on a ship, that follows the underwater vehicle at short range. The AUV can determine its absolute position through bidirectional communication with the mother ship. The accuracy of this system improves with the expansion of the baseline[1], which is dependent on the size of the boat.
- **Ultra-Short Baseline (USBL)**: USBL is one of the most common method, similar to SBL. Transducers are integrated into a single transceiver assembly or an array of transducer elements, called *head transceiver* and placed on the boat. The transducers are positioned, maintaining an approximate distance of 10 centimeters from each other. Distances are calculated with the TOF as in SBL, while the bearing is calculated based on the difference in the phase of the signal arriving at the transceivers. This method allows for the localization of vehicles up to a maximum distance of 200 meters from the head, resulting very accurate in reduced spaces.
- **Long Baseline (LBL)**: At least three transceivers are placed on the seafloor. The vehicle's position is obtained by triangulating the acoustic signals detected by the transponders with the required precision.
- **Long & Ultra Short Baseline (LUSBL)**: It is a variant of the USBL system, merging aspects of both USBL and LBL methodologies. It uses USBL hardware configured in a manner similar to the LBL system, while measurements of range and bearing adhere to the methodologies used in USBL.
- **Single Fixed Beacon**: This method uses a single stationary acoustic beacon as a reference point for navigation. It is known also as Virtual LBL, because it simulates a baseline by projecting the ranges from a single beacon over time until the next update is received.[2]

[1] A *baseline* refers to the distance between two or more reference points, typically transducers or acoustic beacons.

- **Acoustic Modem**: Transducers are not fixed but placed on autonomous non stationary vehicles. The receiver obtains the position of the transmitter along with the communicated data, so it can measures the distance to the transmitter. It can then confine its own position to the surface of a sphere, with the transmitter's location as the center of that sphere. This method is useful for inter-AUV communication for cooperation.



(a) (b) (c)

Figure 1.3: Acoustic position systems: (a) SBL, (b) USBL, (c) LBL

1.2.3 Geophysical sensors

Geophysical navigation systems utilize Earth's natural physical properties to guide and position underwater vehicles, particularly in environments where traditional satellite-based systems are ineffective.
This type og navigation includes sensors from the following categories:

- **Magnetic**: Magnetic sensors operate by detecting changes in magnetic fields for localization. The most common types used in navigation are magnetometers, which can sense the strength and direction of magnetic fields.
- **Acoustic**: Acoustic sensors acoustically detect features in the environment, used as reference points for the navigation. An example of these sensors is given by SONAR.
- **Optical**: This technique involves cameras to capture images and create a real-time map to navigate.

SONAR

Sonar is a sensor that utilizes underwater sound propagation properties to detect and locate objects or even send data, calculating the distance traveled by sound waves based on the speed of sound in water. This method is very effective for mapping large areas of the seabed.
The technique of using sound to navigate and locate objects derives from the natural

world, since some animals like bats and whales utilize echolocation.
There are two types of Sonar:

- **Passive Sonar**: This type does not emit sound waves but listens for sounds made by other objects. It's often used in military applications for detection of submarines or ships.

- **Active Sonar**: This type involves transmitting sound waves and receiving their echoes reflected from objects, through the backscattering phenomenon. When a wave hits a surface or an object, a part of the wave's energy is absorbed, while the rest may be reflected in various directions. The portion of the wave that bounces back towards the source is known as backscatter. By analyzing how waves are backscattered, depending on the properties of the object, like size, shape or composition, it's possible not only localizate objects but also their characteristics. These method requires sophisticated signal processing techniques and a deep understanding of wave physics. Moreover, reflected waves can be low intensity, especially echoes from object with irregular surfaces, which absorb more energy.
  This sensor is widely used for navigation, obstacle avoidance, and object detection. It's possible differentiate between *Imaging Sonars* and *Ranging Sonars*. The first type produces an image of the underwater environment based on the echoes that bounce back from objects. Instead, Ranging Sonars are used for determining water depth (bathymetry), navigation, and avoiding underwater obstacles.

Cameras

Optical navigation systems capture images of the environment and use these visual data to determine the position and orientation of the vehicle. They employ techniques like image processing, pattern recognition, and computer vision algorithms to analyze and interpret the visual data. In particular, the technique used for estimating pose of a robot by analyzing sequences of images captured by a camera, is called *Visual odometry*. This can be achieved through *optical flow*, which calculates the motion of pixels between consecutive video frames, or *structure from motion*, which reconstructes a 3D structure based on the observed motion.[2]

It is appropriate to make a distinction based on the type of cameras used for capturing images:[8]

- **Monocular Camera**: This type of camera uses an only lens to capture images. It is the most common and easy to use. It can't directly perceive the depth of a scene with just a single 2D image. It requires two time successive frames for 3D information extraction and three frames for motion estimation.

- **Stereo Camera**: A stereo camera consists of two or more lenses, mimicking human binocular vision and allowing the camera to capture two slightly different images of the same scene. So by comparing the differences between the images captured by each lens, it can calculate the distance to objects in the scene, creating a 3D perception. It requires two time successive frames for motion estimation.

Although, optical systems have some limitations, like the reduced operational range of cameras, high susceptibility to scattering phenomena and the lighting conditions of the underwater environment. In addition to this, these methods depend on the presence of distinguishable features in the environment. In homogenous underwater areas, they struggle to provide accurate navigational data.

1.3 Virtual simulation of marine environments

Digital Twins and photogrammetry represent essential technologies in marine robotics, providing advanced tools for simulation, mapping, monitoring, and analysis, increasing safety of underwater operations. Photogrammetry provides high-resolution spatial data, which is essential for creating accurate and detailed digital twins of underwater environments. This data is used to build 3D models that reflect the current condition of the physical world. Digital twins enable the simulation of scenarios in a controlled virtual environment, assessing risks and optimizing strategies before deployment in the real world.
Underwater vehicles equipped with high-quality cameras and sensors can collect data about underwater assets. After the data collection, photogrammetry softwares can process it to generate 2D orthophoto visualizations or 3D point cloud models to create digital twins.

However, there are significant challenges in the process of underwater image acquisition. Water affects the intensity of sunlight, reducing visibility and image clarity. This effect is more pronounced with increasing depth and in turbid waters.[13]
Oceans can be dived into three main zones, based on the intenisty of sunlight: the euphotic zone, the disphotic zone and the aphotic zone. The first one is the upper layer of oceans characterized by sufficient sunlight to support photosynthesis. The depth of the euphotic zone can vary, depending on water clarity and geographical location. In the tropics it can reach depths of up to 80 meters, while in polar regions, where sunlight penetration is less, this zone may be limited to a depth of less than 10 meters. In turbid or muddy waters, the euphotic zone can be restricted to just a few centimeters deep.
The disphotic zone is located below the euphotic zone and above the aphotic zone. In this region the sunlight is reduced and there is a gradual transition from light to darkness. In conditions of clear water, this zone can extend to depths of up to 800

meters.
The aphotic zone is characterized by the complete absence of visible sunlight.
Moreover, colors penetrate in different depths based on wavelengths, as shown in figure 1.4. Longer wavelengths with lower energy, like red ones, are absorbed faster than shorter wavelengths and higher energy, like blue. At a depth of 40 meters in saltwater, almost all red light has been absorbed, while blue light continues to penetrate in depth. [14]

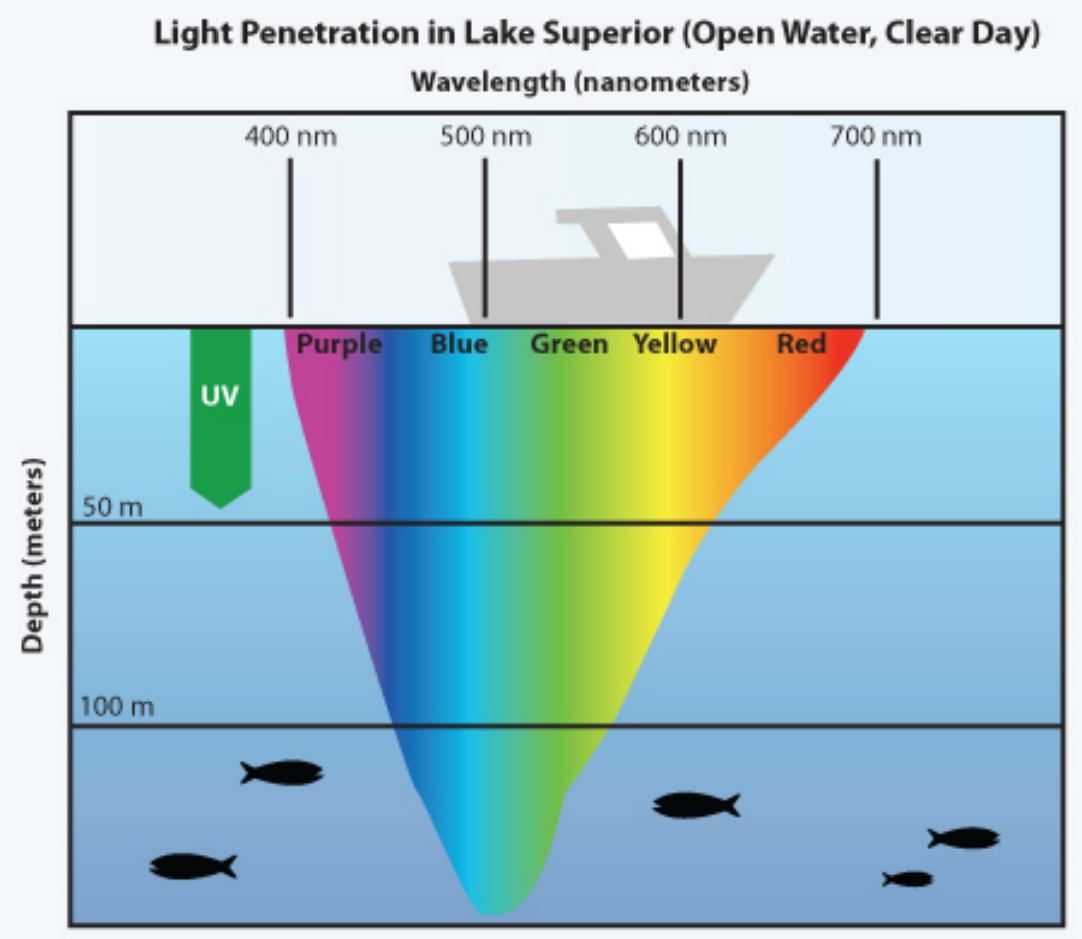


Figure 1.4: How colors penetrate in different ways in oceans. (Image credit: University of Minnesota Sea Grant Program)

Thus, at this depth, without artificial lighting, the environment is perceived in shades of blue. For this reason, it is essential to include a lighting subsystem that emits white light to the underwater environment.
Images captured with artificial lights are not composed only of *direct light* but there is a *backscatter* component of light that has not interacted with the object and a *blur* component that has been reflected from the target. The backscattered light component affect the camera even without interacting with the object, due to suspended particles in the water, creating unwanted contrast differences and mask

details in the scene. So, if the lighting is not appropriate, it can become a disturbing factor itself for image acquisition in water.[15]
In addition to this, color and reflectivity of objects vary when captured from varying camera perspectives and distances, causing problems for algorithms in matching and aligning image data. This is because Brightness Constancy Constraint (BCC) is violated by a moving camera.[16]
In [17] literature review on light projection and light-sensing technologies is presented.

1.3.1 Photogrammetry

Photogrammetry is a technology used for measuring information about physical objects and their environments by reconstructing a 3D model, often a point cloud, from 2D digital images. The first usage of this technology dates back to 1867 when the architect Albrecht Meydenbauer used it to document a church building in Germany.[18] Since then, its application has expanded across numerous fields, from archaeology, as shown in [19], to subsea applications, in [20].
A standard photogrammetry system consists of cameras and retro-reflective targets and a preparation, like software configuration, strategic camera placement and the final step of 3D reconstruction. The careful positioning of cameras and the extensive coverage of the object surfaces are essential to ensure the accuracy and precision of measurements.[21]

Calibration

Calibration in photogrammetry involves adjusting and fine-tuning the parameters of cameras to ensure accuracy in the captured images. This process is essential because accurate 3D models and measurements depend on the precision of the initial images. Calibration ensures that the camera's lens distortions, focal length, and other intrinsic and extrinsic parameters are correctly configured.
The calibration process involves two critical steps. Firstly, the camera's intrinsic parameters are computed, which include aspects of the lens such as focal length, optical center, and lens distortion. The second step determines the position and orientation of the camera, known as the extrinsic parameters. This is crucial for aligning the camera in relation to a laser projector in laser triangulation systems, or in relation to another camera in stereo vision setups.[17]
Camera calibration is a significant issue associated with underwater imaging, due to the refraction caused by the air-water interface. This refraction occurs due to the difference in density between the two media. This phenomenon must be accurately accounted for to ensure precise calibration and to mitigate the effects of distortion in underwater imaging. [22]

Photo-mosaic

Photo mosaics in photogrammetry involve the process of stitching together multiple overlapping photographs to create a single image or map. The creation of a photo mosaic begins with capturing a series of overlapping images of the target area. Then, there is image stitching process, which includes the following steps:

1. **Feature extraction and matching**: The first step in the panoramic recognition algorithm is to extract and match SIFT features between all the images. Once features have been extracted from all n images, they must be matched to its k nearest neighbors in feature space.

2. **Image matching**: The second step is finding all matching images, so connected sets of image matches create a panorama. From the feature matching step, images that have a large number of matches between them, has been identified. Now, a constant number m of images, that have the greatest number of feature matches to the current image, is considered as potential image match.

3. **Image fusion**: The last step combines the information of matched images and bends them into a single image to create a panorama. This part of the process often involves color correction and balancing to ensure uniformity across the mosaic, especially in cases where lighting conditions may vary across different photographs.

So, it is required that images have a partial overlap. If this overlap is not present due to low-quality images or not rich environments, it is possible to refer to navigation data from acoustic positioning sensors installed on underwater vehicles to estimate the vehicle's trajectory.
Another problem consists in identifying a planar transformation to align two or more 2D images taken from different perspectives. There are two methods to solve this problem:

- **Direct methods**: Direct methods, also known as featureless methods, depend on optimizing the photometric consistency across overlapping areas of the images. These methods are effective for minor translations and rotations. However, in underwater imaging, images are often captured using stroboscopic lighting due to power constraints that affect the vehicle's autonomy. This leads to low-frequency image acquisition and results in insufficient overlap for registration through direct methods.

- **Feature-based methods**: These methods utilize a sparse set of points and correlations between image pairs to determine the transformation between them. Feature-based methods consist of two phases. Initially, in the feature detection stage, it's crucial to identify points of interest in images. Then, the feature matching stage associates these identified points based on a specific

descriptor.
This can be obtained through two different strategies. The first one involves using a feature detector algorithm to identify salient points in one image, then recognizing these features in another image. Identification in this approach is performed using cross-correlation or a sum of squared differences measure, focusing on the pixel values around the interest point.
A second method detects interest points on both images using invariant image descriptors. The correspondence problem is solved by comparing their descriptor vectors. These descriptors are invariant to geometrical and photometric transformations between the image pairs. This robustness is valuable in underwater imaging.

After solving the correspondence problem of two images, the correspondence set found can be utilized to calculate a planar transformation that describes the camera's movement between the two images. This transformation is represented by a homography matrix H, which can describe motion across up to eight Degrees Of Freedom (DOF). The accuracy of the homography depends on the quality of the correspondences found for its calculation. Non-uniform illumination, shadows and digital noise, can produce matching failures. Additionally, moving objects might induce correspondences that don't align with the dominant motion between the two images. These are known as outliers. Therefore, an algorithm have to discern correct and incorrect correspondences.
Blending algorithms are divided into two primary groups based on their working principle: *transition smoothing* method, that reduces the visibility of the junction areas between images by combining the overlapping information, and *optimal seam finding* methods, that seeks to place the seam in the least noticeable area, identifying the best path for cutting the images where the photometric differences between them are minimal. The second method is adapt for moving objects because it avoids overlapping areas with noticeable movement. The combination of these two methods leads to the development of *hybrid techniques*.

In addition to optical information, bathymetric information from acoustic sensor can be integrated, obtaining the 2.5D mosaic. [23]

3D Mapping

3D mapping is a technique which captures a series of overlapping photographs in underwater settings, then used to create detailed 3D models of the landscapes and structures. In 2D and 2.5D mapping, it is assumed that there is a base plane into which images can be projected. Once the motion is extracted, the mapping process primarily involves warping and deforming the images into this plane. However, in 3D mapping, this planar base is absent, making the task more complex.
In both acoustic and optical sensors, reconstructing the shape of an object requires

the vehicle's trajectory. For acoustic data, the 2D swath from a multibeam sensor, combined with the vehicle's motion, can generate a 3D point cloud. For optical data, once the camera's trajectory is known, 3D positions of points can be triangulated using the line-of-sight rays from each camera–2D feature pair.
So, the mapping problem is related to trajectory estimation. Structure from Motion (SfM) methods are used to solve this problem when dealing with purely optical data, and more generic Simultaneous Localization and Mapping (SLAM) methods when additional information is available.

Structure from Motion involves three main steps:

- **Rectification**: This initial step involves transforming each image so that pairs of conjugate epipolar lines become collinear and parallel to the horizontal image axis. This adjustment simplifies the correspondence problem from a 2D search to a more manageable 1D search.
- **Correspondence search**: In this phase, the correspondence between pixels in the left and right images is established. For a given pixel in the left image, the corresponding pixel is searched in the same row of the right image.
- **Reconstruction**: The final step uses a triangulation algorithm. By inputting each pixel and its corresponding match, the 3D position of that pixel can be computed.

Structure from Motion solves two problems: surveying an unknown structure from known camera positions and determining camera motion based on known fixed points. So, SfM is a versatile technique that combines photogrammetry and computer vision, applied in various field, as shown in [24].

1.3.2 Digital Twin

A digital twin is a virtual replica of a physical entity, process, or system that can be used for various purposes, including simulation, monitoring, and predictive analysis. By simulating different scenarios, operators can plan and test responses to potential emergencies or system failures without the risks associated with physical trials.
Moreover, digital twins play a crucial role in environmental conservation. They can be used to model marine ecosystems, allowing researchers to study the impact of climate change, pollution, and human activities on these environments.[25]

The concept of "digital twin" first apperared during NASA's Apollo 13 program in the 1960s. A digital twin model of the Apollo 13 spacecraft was created on Earth, enabling engineers on the ground to simulate and test potential solutions for the rescue mission in space.
Michael Grieves provided the definition of the digital twin concept in his presentation about a product life cycle management in 2003.

A digital twin consists of five dimensions, that interact to create a comprehensive digital representation of a physical entity.

- **A physical part**: The physical part represents the real-world entity, such as a building, infrastructures or environments.
- **A virtual part**: The virtual part is a mirror of the physical part, created in a virtual environment.
- **Connections**: The connection aspect facilitates data transfer between the physical and virtual entities. The connection typically requires data transfer from the physical to the virtual entity, feedback in the reverse direction is not mandatory.
- **Data**: Data dimension involves the storage and management of information gathered from the physical entity.
- **Services**: The digital twin must provide services like health monitoring and decision-making support. These services are based on the analysis of data collected from the physical entity.

Digital twins have to face up to two challenges: creating a high-fidelity virtual representation of the physical object, and rapidly updating the collected data for the digital twin's diagnosis and decision-making.
Regarding the first one, researchers have developed photogrammetry-based reconstruction workflows. However, the use of digital twins as a continuous monitoring framework is still relatively unexplored and represents a significant opportunity for advancement in this field.[18]
In [26] technologies for digital twin are shown, which include data-driven, statistical or machine learning strategies.

Digital twins have found applications in large sectors, from manufacturing industry, supply chain management and preventive maintenance, to agriculture, healthcare or weather modeling and many more, as demonstrated in [27].

Chapter 2

Mathematical model for design and control

This chapter offers a detailed analysis of the kinematic and dynamic model for a ROV. Through the application of principles in mechanics and fluid dynamics, the model aims to capture the essential aspects of ROV motion and behavior. A marine craft subjected to the forces of wind, waves, and ocean currents experiences motion in six DOF. To model these movements accurately, the equations of motion are typically derived through the Newton–Euler or Lagrange laws.
For the composition of this chapter, the methodology outlined in [28] and in [29] has been adhered to.

In order to simplify the analysis of the mathematical model, some assumptions have been introduced as follows:

1. It is assumed that the water is an ideal fluid, characterized by being incompressible, non-viscous and non-rotational.
2. The fixed ground reference system is considered inertial.
3. The ROV is considered as a rigid body that is completely submerged in water.
4. The ROV's speed is very low (less than 2 m/s) so lift forces can be excluded.
5. Wave-induced disturbance is neglected, as the ROV is fully submerged.
6. The dynamics of the tether connected to the ROV are not modeled.
7. The ROV is considered to have symmetry in both the xz and xy planes, with the center of gravity located within these planes of symmetry.
8. Assuming that the ROV operates below the wave-affected zone, the impact of wave disturbances on the vehicle can be excluded.
9. The ROV has 4 DOF. The thrusters position of the ROV used in this thesis does not allow for active control of the pitch and roll orientation. However, the motion around these angles is considered self-regulated due to the restoring moments of the vehicle's buoyancy.

It's necessary to define how the reference frames are oriented for this modeling. Two fixed frames are considered: *the Earth-fixed frame*, with X, Y, Z axes and the origin in the center of the Earth, and *the Body-fixed frame*, with X_b, Y_b, Z_B axes and the origin in the geometrical centre of the vehicle, oriented as in figure 2.1.

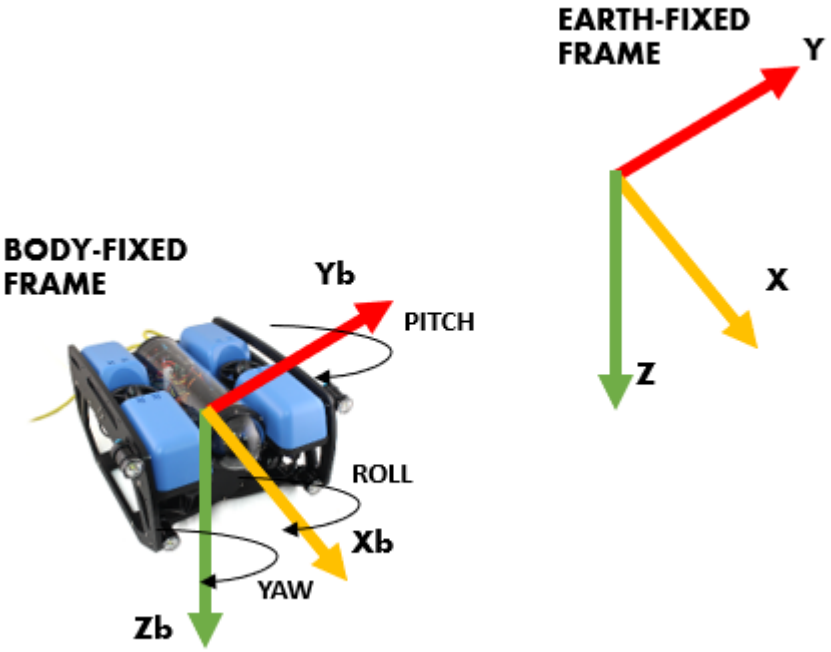


Figure 2.1: Earth-fixed and Body-fixed frames

In addition to these frames, it is appropriate to define the North-East-Down coordinate system (NED), used in everyday life. This system has the origin on the Earth's reference ellipsoid and this system's x-axis points towards true North, the y-axis towards the East, and the z-axis downwards, perpendicular to the Earth's surface. The position of this frame relative to the Earth-fixed frame is defined using longitude and latitude coordinates. However, the flat Earth navigation is considered, assuming that the area of operation is relatively flat and that longitude and latitude remain approximately constant. In this context, a tangent plane to the Earth's surface is used as the reference system for navigation and it is considered inertial to simplify navigation calculations.

2.1 Kinematic model

A vehicle that can freely move in the space has up to six DOF, three translational and three rotational. Therefore, a marine vehicle with full actuation in all 6 DOFs requires actuators capable of generating independent forces and moments in every direction.

To simulate the motion of such a craft accurately, it is essential to utilize a set of 12 ordinary differential equations. Horizontal motion consists of surge (longitudinal) and sway (sideways), while yaw describes rotation about the vertical axis (heading). The other three DOFs are roll (longitudinal axis rotation), pitch (transverse axis rotation), and heave (vertical motion). In marine applications, yaw is often the primary focus for feedback control, with stationkeeping involving stabilization of surge, sway, and yaw.
In 2.1 the used Society of Naval Architects and Marine Engineers (SNAME) nomenclature for position, velocity, and forces is presented. A vectorial representation is

Movement	**Name**	**Position**	**Velocity**	**Force/Moment**
X translation	Surge	x	u	X
Y translation	Sway	y	v	Y
Z translation	Heave	z	w	Z
X rotation	Roll	ϕ	p	K
Y rotation	Pitch	θ	q	M
Z rotation	Yaw	ψ	r	N

Table 2.1: Notation used in underwater vehicle dynamics, detailing movements, positions, velocities, and forces.

used for positions, velocities and forces, defined as follow:

$$\eta = (x, y, z, \phi, \theta, \psi)^T \tag{2.1}$$

$$\nu = (u, v, w, p, q, r)^T \tag{2.2}$$

$$\tau = (X, Y, Z, K, M, N)^T \tag{2.3}$$

where η is the vector of the vehicle's positions respect the Earth-fixed frame, ν is the vector of vehicle's velocities respect the Body-fixed frame and τ is the vector of vehicle's forces and moments respect the Body-fixed frame.
ϕ and ψ are defined into the interval $[-\pi, \pi)$, while θ into the interval $(-\frac{\pi}{2}, \frac{\pi}{2})$ due to the singularity of the rotational matrix, shown later in the chapter.

It is possible to decompose the previous vectors into two vectors, represented the linear variables and the angular ones:

$$\eta = \begin{bmatrix} P \\ \Theta \end{bmatrix}, \quad \nu = \begin{bmatrix} v \\ \omega \end{bmatrix}, \quad \tau = \begin{bmatrix} f \\ m \end{bmatrix} \tag{2.4}$$

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3, \quad \Theta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathbb{R}^3, \quad v = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3, \quad \omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3, \quad f = \begin{bmatrix} X \\ Y \\ X \end{bmatrix} \in \mathbb{R}^3, \quad m = \begin{bmatrix} K \\ M \\ N \end{bmatrix} \tag{2.5}$$

where:

- **P** is the linear positions vector
- Θ is the angular positions vector
- **v** is the linear velocities vector
- ω is the angular velocities vector
- **f** is the vector of forces exerted on the vehicle
- **m** is the vector of moments exerted on the vehicle

To convert variables from the Body-fixed frame n to the Earth-fixed frame b, the rotational matrix $R_b^n(\Theta)$ must be used as follow:

$$v_n = R_b^n(\Theta) v_b \tag{2.6}$$

where v_n is the linear velocities vector in the Earth-fixed frame and v_b in the Body-fixed frame.
The rotational matrix $R_b^n(\Theta)$ is calculated as follow:

$$R_b^n(\Theta) = R_z(\psi) R_y(\theta) R_x(\phi) \tag{2.7}$$

with:

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.9}$$

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.10}$$

Therefore, the extended rotation matrix $R_b^n(\Theta)$ is:

$$R_b^n(\Theta) = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\phi\sin\theta\sin\psi & -\cos\psi\sin\phi + \sin\theta\sin\psi\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.11}$$

This matrix satisfies the properties: $RR^T = R^T R = I$ and $\det(R) = 1 \Rightarrow R$ is orthonormal. Consequently, the inverse rotation matrix is given by: $R^{-1} = R^T$.

Similarly, the transformation of angular velocities is given by:

$$\dot{\Theta} = T_\Theta(\Theta)\omega_b \tag{2.12}$$

where ω_b and $\dot{\Theta}$ are the angular velocities in the Body frame and in the Earth frame, respectively.
The transformation matrix $T_\Theta(\Theta)$ is derived as follows:

$$T_\Theta(\Theta) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \tag{2.13}$$

This matrix has a singularity when the pitch angle is $\theta = \pm\frac{\pi}{2}$. However, the ROV rarely approaches these pitch angles because it has a weight distribution such that the center of gravity is lower than the center of buoyancy, thus the vehicle is self-balanced around the neutral pitch angle $\theta = 0$. Otherwise, the singularity can be avoided by using quaternions[1].

The 6 DOF kinematic equation can be written in vector form as:

$$\dot{\eta} = J(\eta)\nu \iff \begin{bmatrix} \dot{p} \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} R_b^n(\Theta) & 0_{3x3} \\ 0_{3x3} & T_\Theta(\Theta) \end{bmatrix} \begin{bmatrix} v_b \\ \omega_b \end{bmatrix} \tag{2.14}$$

2.2 Hydrodynamic model

The Hydrodynamic model of an underwater vehicle can be described through Newton–Euler equations, as follow:

$$\mathbf{M}\dot{\nu} + \mathbf{C}(\nu)\nu + \mathbf{D}(\nu)\nu + \mathbf{g}(\eta) = \boldsymbol{\tau} + \boldsymbol{\tau}_d \tag{2.15}$$

$$\boldsymbol{\tau} = \mathbf{B}_t\mathbf{u}_t \tag{2.16}$$

where:

[1] It is a four-parameter method based on unit quaternions. A quaternion q is defined as a complex number with one real part and three imaginary parts.

- $\mathbf{M} \in \mathbb{R}^{6x6}$ is the matrix of inertia and added mass.
- $\mathbf{C} \in \mathbb{R}^{6x6}$ is the centripetal and Coriolis matrix.
- $\mathbf{D} \in \mathbb{R}^{6x6}$ is the hydrodynamic damping matrix.
- $\mathbf{g} \in \mathbb{R}^{6x1}$ is the vector of gravitational and buoyancy forces.
- $\mathbf{B}_t \in \mathbb{R}^{6x6}$ is the thrusters allocation matrix.
- $\mathbf{u}_t \in \mathbb{R}^{6x1}$ is the vector containing the force generated by the thrusters.
- $\boldsymbol{\tau} \in \mathbb{R}^{6x1}$ is the vector of forces and moments applied to the vehicle.
- $\boldsymbol{\tau}_d \in \mathbb{R}^{6x1}$ represents environmental disturbances.

2.2.1 Inertial matrix and added mass

The $\mathbf{M}$ matrix represents the force and moment due to the acceleration of the ROV (rigid body mass) and water (added mass) around the vehicle, calculated as:

$$\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A \tag{2.17}$$

The rigid body mass matrix $\mathbf{M}_{RB}$ is defined as follows:

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z \end{bmatrix} \tag{2.18}$$

where:

- m is the mass of the vehicle.
- I_i is the inertial moment of the i axis.
- I_{ij} is the inertial product on the ij plane, measuring the imbalance of mass distribution (with $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$, $I_{yz} = I_{zy}$).
- $\mathbf{r}_g := [x_g, y_g, z_g]^T$ is the Center of Gravity.
- $\mathbf{r}_b := [x_b, y_b, z_b]^T$ is the Center of Buoyancy.

Particularly, in a ROV, the origin of the vehicle-fixed reference system o_b is placed at the geometric center of the ROV. Assuming that the Center of Buoyancy coincides with the origin o_b implies: $x_b = 0, y_b = 0, z_b = 0$. The vehicle has symmetry in the xz and xy planes but the Center of Gravity in z_g may be placed lower than the origin

o_b so that: $x_g = 0, y_g = 0, z_g \neq 0$ and $I_{xy} = I_{xz} = I_{yz} = 0$. $\mathbf{M}_{RB}$ can be simplified as follow:

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_g & 0 \\ 0 & m & 0 & -mz_g & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & -mz_g & 0 & I_x & 0 & 0 \\ mz_g & 0 & 0 & 0 & I_y & 0 \\ 0 & 0 & 0 & 0 & 0 & I_z \end{bmatrix} \tag{2.19}$$

The added mass matrix $\mathbf{M}_A$ can be derived using an energy-based approach according to Kirchhoff's equation. It is defined as follows:

$$\mathbf{M}_A = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{2.20}$$

This matrix is symmetric, so $\mathbf{M}_A = \mathbf{M}_A^T$. The hydrodynamic derivatives are represented using the SNAME notation. For example, the hydrodynamic derivative $Z_{\dot{u}}$ is the added hydrodynamic mass force Z in the z-direction (heave) due to an acceleration $\dot{u}$ along the x-axis (surge), expressed as:

$$Z_{\dot{u}} = \frac{\partial Z}{\partial u} \tag{2.21}$$

Also $\mathbf{M}_A$ can be simplified, as the movements between the degrees of freedom of the ROV in hydrodynamics are assumed to be decoupled:

$$\mathbf{M}_A = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix} \tag{2.22}$$

This matrix includes the force and moment due to the acceleration of the fluid around the ROV.

2.2.2 Centripetal and Coriolis matrix

The Coriolis force matrix can also be decomposed into a term concerning the rigid body and a term concerning the added mass, similarly to before:

$$C(\nu) = C_{RB}(\nu) + C_A(\nu) \tag{2.23}$$

In particular:

$$CRB(\nu) = \begin{bmatrix} 0 & 0 & 0 & mz_g r & mw & -mv \\ 0 & 0 & 0 & -mw & mz_g r & mu \\ 0 & 0 & 0 & -mz_g p + mv & -mz_g q - mu & 0 \\ -mz_g r & mw & mz_g p - mv & 0 & I_{zr} & -I_{yq} \\ -mw & -m_{zg} r & m_{zg} q + mu & -I_z r & 0 & I_x p \\ mv & -mu & 0 & I_y q & -I_x p & 0 \end{bmatrix} \tag{2.24}$$

$$CA(\nu) = \begin{bmatrix} 0 & 0 & 0 & 0 & Z_{\dot{w}} w & 0 \\ 0 & 0 & 0 & -Z_{\dot{w}} w & 0 & -X_{\dot{u}} u \\ 0 & 0 & 0 & -Y_{\dot{v}} v & X_{\dot{u}} u & 0 \\ 0 & -Z_{\dot{w}} w & Y_{\dot{v}} v & 0 & -N_{\dot{r}} r & M_{\dot{q}} q \\ Z_{\dot{w}} w & 0 & -X_{\dot{u}} u & N_{\dot{r}} r & 0 & -K_{\dot{p}} p \\ -Y_{\dot{v}} v & X_{\dot{u}} u & 0 & -M_{\dot{q}} q & K_{\dot{p}} p & 0 \end{bmatrix} \tag{2.25}$$

Where $X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}$ are the added mass coefficients of the Coriolis forces.

2.2.3 Hydrodynamic damping matrix

There are four main sources that cause hydrodynamic damping for a marine vehicle, including: potential damping, wave drift damping, skin friction, and damping due to vortex shedding. However, the effects of potential damping and wave drift damping are omitted for underwater vehicles, because these vehicles operate mostly below the surface.

To simplify the model of the system, a rough estimate is that the higher order terms beyond the second can be ignored, since the ROV has three planes of symmetry and performs uncoupled movements.

So, the hydrodynamic damping matrix can be decomposed into a term representing the skin friction (linear) and another one representing the damping due to vortex shedding (non linear), as:

$$D = D_l + D_{nl} \tag{2.26}$$

In particular:

$$D_l = \begin{bmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & Y_p & 0 & Y_r \\ 0 & 0 & Z_w & 0 & Z_q & 0 \\ 0 & K_v & 0 & K_p & 0 & N_r \\ 0 & 0 & M_w & 0 & M_q & 0 \\ 0 & N_v & 0 & N_p & 0 & N_r \end{bmatrix} \tag{2.27}$$

where X_u, Y_v, Z_w, K_p, M_q, N_r are the linear hydrodynamic damping coefficients.
It can be simplified as:

$$D_l = \text{diag}(X_u, Y_v, Z_w, K_p, M_q, N_r) \tag{2.28}$$

$$D_{nl} = \text{diag}(X_{|u|u}|u|, Y_{|v|v}|v|, Z_{|w|w}|w|, K_{|p|p}|p|, M_{|q|q}|q|, N_{|r|r}|r|) \tag{2.29}$$

where $X_{|u|u}$, $Y_{|v|v}$, $Z_{|w|w}$, $K_{|p|p}$, $M_{|q|q}$, $N_{|r|r}$ are the nonlinear hydrodynamic damping coefficients.
Thus, the total matrix is defined as:

$$D = \begin{bmatrix} X_u + X_{|u|u}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v + Y_{|v|v}|v| & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_w + Z_{|w|w}|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_p + K_{|p|p}|p| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_q + M_{|q|q}|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_r + N_{|r|r}|r| \end{bmatrix} \tag{2.30}$$

2.2.4 Vector of gravitational and buoyancy force

The restoring force $g(\eta)$ is the net buoyancy, where $W = mg$ is the weight of the ROV, $B = \rho g V$ is the buoyancy. Where:

- ρ is the density of water.
- g is the gravitational acceleration.
- V is the volume of fluid displaced by the ROV.

In most cases, the robot is positively buoyant ($W < B$) so that the vehicle can rise to the surface if propulsion is lost.
The vector is defined as follow:

$$g(\eta) = \begin{bmatrix} (W - B)\sin\theta \\ -(W - B)\cos\theta\sin\phi \\ -(W - B)\cos\theta\cos\phi \\ -(y_g W - y_b B)\cos\theta\cos\phi + (z_g W - z_b B)\cos\theta\sin\phi \\ -(z_g W - z_b B)\sin\theta + (z_g W - z_b B)\cos\theta\cos\phi \\ -(x_g W - x_b B)\cos\theta\sin\phi - (y_g W - y_b B)\sin\theta \end{bmatrix} \tag{2.31}$$

2.2.5 Thrusters allocation matrix and forces

It is considered a vehicle with eight propellers and controllable in six DOF.
The control force due to thrusters can be defined by the following equation:

$$F = Ku \tag{2.32}$$

where $\mathbf{u} = [u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8]^T$ whose elements u_i are the control inputs of each thruster and $K = \text{diag}[K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8]^T$ whose elements K_i are the thrust coefficients, which are scalar factors. The force vector can be represented

by $F = [F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8]^T$.

The forces and moments in 6 DOF can be determined by:

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{f} \\ \mathbf{r} \times \mathbf{f} \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \\ F_z l_y - F_y l_z \\ F_x l_z - F_z l_x \\ F_y l_x - F_x l_y \end{bmatrix} \tag{2.33}$$

where $\mathbf{f} = [F_x, F_y, F_z]^T$ is the force vector and $\mathbf{r} = [l_x, l_y, l_z]^T$ is the moment arms vector.
2.33 can be defined as follow:

$$\boldsymbol{\tau} = T(\alpha)F = T(\alpha)Ku \tag{2.34}$$

where $T = [t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8]^T \in \mathbb{R}^{6x8}$ is the thrust configuration matrix and $\alpha \in \mathbb{R}^8$ is the vector of azimuth angles.

The control allocation method defines the control input signal $\mathbf{u}$ to be applied to the thrusters in order to reach the desired forces $\boldsymbol{\tau}$.
By inverting Equation 2.34, it is possible to find $\mathbf{u}$, as follows:

$$\mathbf{u} = K^{-1}T^{-1}(\alpha)\boldsymbol{\tau} \tag{2.35}$$

However, the thrust configuration matrix $T(\alpha)$ is not a square matrix for this application, so it is necessary to calculate its pseudo-inverse $T^+(\alpha)$ through the Moore-Penrose inverse method:

$$T^+(\alpha) = T(\alpha)^T(T(\alpha)T(\alpha)^T)^{-1} \tag{2.36}$$

Therefore, the control input is defined as:

$$\mathbf{u} = K^{-1}T(\alpha)^+\boldsymbol{\tau} \tag{2.37}$$

Finally, the 2.15 equation of nonlinear dynamics of the ROV with a generalized disturbance $\boldsymbol{\tau}_d$ can be rewritten as:

$$\mathbf{M}\dot{\nu} + \mathbf{C}(\nu)\nu + \mathbf{D}(\nu)\nu + \mathbf{g}(\eta) = \mathbf{K}_p(\mathbf{A}\mathbf{u}) + \boldsymbol{\tau}_d \tag{2.38}$$

2.3 Parameters of a BlueROV2

In particular, a BlueROV2, detailed in the next chapter, was used for this project. Thus, in table 2.2, the moment arms of the 8 thrusters relative to the Center of Gravity of the BlueROV2 are calculated. The rotation angles of the horizontal thrusters

Thrust	l_{xi} (m)	l_{yi} (m)	l_{zi} (m)
T1	0.156	0.111	0.085
T2	0.156	-0.111	0.085
T3	-0.156	0.111	0.085
T4	-0.156	-0.111	0.085
T5	0.120	0.218	0
T6	0.120	-0.218	0
T7	-0.120	0.218	0
T8	-0.120	-0.218	0

Table 2.2: Moment arms of the 8 thrusters of the BlueROV2.

from T1 to T4 are respectively: $\pi/4$, $-\pi/4$, $-3\pi/4$, and $3\pi/4$. The thrusters from T5 to T8 are vertical thrusters without horizontal rotations.

The physical and hydrodynamic parameters of the BlueROV2 are summarized in Table 2.3.

2.4 State-space model

The state of the system $\mathbf{x} \in \mathbb{R}^{12}$ is selected as the position and velocity of the vehicle, defined as follow:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\eta} \\ \nu \end{bmatrix} \tag{2.39}$$

Due to the non linearity in both the dynamics and the kinematics of the ROV, a nonlinear state-space model is used to represent the dynamic positioning controller model:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\boldsymbol{\eta}} \\ \dot{\nu} \end{bmatrix} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\tau}_d, t) = \begin{bmatrix} \mathbf{J}(\boldsymbol{\eta})\nu \\ \mathbf{M}^{-1}[\mathbf{K}_p(\mathbf{A}\mathbf{u}) + \boldsymbol{\tau}_d - \mathbf{C}(\nu)\nu - \mathbf{D}(\nu)\nu - \mathbf{g}(\boldsymbol{\eta})] \end{bmatrix} \tag{2.40}$$

Parameter	**Symbol**	**Value**	**Unit**
Mass	m	11.5	kg
Buoyancy	B	114.80	N
Weight	W	112.82	N
Center of Gravity	$\mathbf{r}_g$	(0,0,0)	m
Center of Buoyancy	$\mathbf{r}_b$	(0,0,0.02)	m
Inertia Moment	I	diag(0,0,0.2)	kg·m^2
Added Mass Parameters	$X_{\dot{u}}$	-5.5	kg
	$Y_{\dot{v}}$	-12.7	kg
	$Z_{\dot{w}}$	-14.57	kg
	$K_{\dot{p}}$	-0.12	kg·m^2/rad
	$M_{\dot{q}}$	-0.12	kg·m^2/rad
	$N_{\dot{r}}$	-0.12	kg·m^2/rad
Linear Damping Parameters	X_u	-4.03	N·s/m
	Y_v	-6.22	N·s/m
	Z_w	-5.18	N·s/m
	K_p	-0.07	N·s/rad
	M_q	-0.07	N·s/rad
	N_r	-0.07	N·s/rad
Non Linear Damping Parameters	$X_{u\|u}$	-18.18	N·s^2/m^2
	$Y_{v\|v\|}$	-21.66	N·s^2/m^2
	$Z_{w\|w\|}$	-36.99	N·s^2/m^2
	$K_{p\|p\|}$	-1.55	N·s^2/rad^2
	$M_{q\|q\|}$	-1.55	N·s^2/rad^2
	$N_{r\|r\|}$	-1.55	N·s^2/rad^2

Table 2.3: BlueROV2 Parameters

Chapter 3

Approach

This chapter provides an analysis of the approach used to reach the project's final purpose, applying the theory and solving some challenges showed in previous chapters. Especially, the chapter will proceed with details of the hardware and software components used, emphasizing the key technical features essential for the project.
The components are divided in *Underwater Setup*, *Surface Setup* and *Software Application*. In the subsection *Final Architecture* it is explained how the hardware and software communicate with each other in the system.
Following this, an examination of the algorithms and programming paradigms of the control systems implemented is presented, pointing out the logic used behind the architecture.
In the final section of this chapter, attention is directed towards an experimental environment developed to simulate a digital twin of the ROV. This innovative approach is centered around the synergistic fusion of the ROV's real-time operational behavior with the advancements in simulation technology, with the final aim of scanning the seabed.

3.1 Materials and Methods

This section is dedicated to present the underwater setup, the surface setup and software applications used for the current project, highlighting its various components and their functionalities. Central to this setup is the ROV and its sensors and instruments for data acquisition and environmental monitoring. The section details the specifications, capabilities, and integration of these devices, including the communication systems, power management, and navigation tools, to explain how they collectively contribute to the system.

3.1.1 Underwater Setup

BlueROV2

For this project a BlueROV2, manufactured by BlueRobotics, was used. This underwater vehicle, in figure 3.1, belongs to ROV category and it is distinguished by

its high maneuverability and stability, facilitated by a vectored thruster configuration that can provides six DOF in movement.

(a) (b)

Figure 3.1: BlueROV2 with Heavy Configuration Retrofit Kit: (a) top view, (b) front view.

The standard model is equipped with six brushless T200 thrusters made of 316 marine grade stainless steel and polycarbonate, four of which vectored and two vertical, allowing the vehicle to have 4 DOF.
The four vectored thrustesrs give the ROV the freedom to move in the directions of sway, surge, and yaw. The other two thrusters facilitate movements in the heave direction. While the arrangement of these thrusters theoretically allows for roll movement, current software constraints prevent this functionality. Presently, the ROV lacks the ability to maneuver in the pitch direction, a limitation that could be overcome only through the integration of two additional thrusters.

Thrusters include encapsulated windings, stator, coated magnets, and rotor. The fully-flooded design is a distinctive feature. This design facilitates water cooling of the motor and water lubrication of plastic bushings, eliminating the need for shaft seals, magnetic couplings, and air or oil filled compartments. Consequently, the thruster exhibits natural pressure tolerance. Operating optimally at a voltage of 16V, the thruster demonstrates versatility by also being able to operate at a vast range of voltages. The thrusters are designed with clockwise and counterclockwise propeller orientations to minimize torque reactions. All the thrusters are controlled by PWM signals and the system's gain levels are adjustable, allowing to have precision control at minimal velocities, as well as the requisite force to overcome aquatic currents.
The configuration can be upgraded to include eight thrusters by using the *Heavy Configuration Retrofit Kit* for a complete control across all six DOF and active pitch and roll feedback stabilization. The kit introduces two additional vertical thrusters and moves all vertical thrusters to the exterior of the ROV's frame, including also

additional buoyancy for more stability and lifting capacity. The Heavy Configuration does not improve the horizontal speed of the BlueROV2 but the vertical thrust, making it advisable when additional accessories, such as sonars or manipulators, are being integrated into the system.

The BlueROV2 is equipped with a set of electronics and sensors, including an HD camera for clear underwater imaging, and advanced lighting systems to enhance visibility in the depths of underwater environments.
The high-definition camera wirh 1080p resolution, 30 frames per second, 200 ms latency, is positioned at the forefront of the ROV, featuring a wide-angle lens and specialized for low-light conditions. This camera's angle can be adjusted upward or downward, maintaining visual clarity even when the ROV is in a horizontally level. This camera is used as an additional sensor in this project.
The artificial lights system can emitt a combined illumination of up to 6,000 lumens.

The BlueROV2 offers configuration options with either acrylic plastic or anodized aluminum enclosures. The first one is designed to reach depths of up to 100 meters, while the anodized aluminum can reach depths of up 300 meters. It is also included a vacuum test pump, used to ensure the proper seal before each dive.

The distinctive feature of the BlueROV2 compared to traditional ROV models, is its modular design, making it adaptable to various mission requirements through a range of attachable sensors and tools, like scanning sonar or navigation systems.

The BlueROV2 is initially set up for operation via Lithium-ion battery power. However, it can be also powered by a high-voltage topside power supply, illustrated in the next section, which eliminates the need for batteries. The latter was adopted in this project.

The BlueROV2 also provides a Navigator Flight Controller (Figure 3.3) and BlueOS. The Navigator is a controller specifically engineered for ROVs, equipped with onboard sensors such as an IMU and a magnetometer. This electronic board is equipped with an advanced processor, STMicroelectronics sensors, and runs on the real-time operating system NuttX. Featuring 16 outputs, the Navigator can connect to a variety of devices including thrusters (controlled through MAVLink), lights, grippers, and other accessories. Additionally, it boasts several serial and I2C communication ports for interfacing with sensors and sonars. The controller includes features like integrated multithreading, a Unix/Linux-like programming environment, autopilot functions for missions and flight behaviors, and a customized PX4 driver level.[30]

Paired with the Navigator is the Raspberry Pi 4 computer (Figure 3.2), responsible for all processing and computing tasks within the ROV. This system runs the open-

source BlueOS software, that operates the ArduSub vehicle control software, oversees the camera and tether connection, and simplifies software updates to integrate new functionalities.

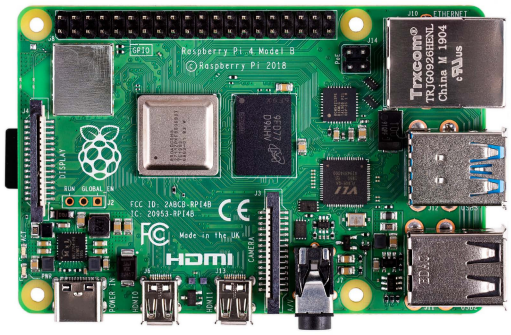


Figure 3.2: Raspberry Pi 4

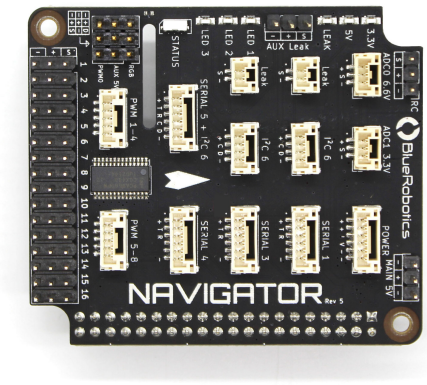


Figure 3.3: Navigator Flight Controller. (Image credit: BlueRobotics)

The Raspberry Pi 4 is connected to the tether and uses Ethernet technology to transmit telemetric data to the surface. It has a 64-bit quad-core processor and supports dual displays with resolutions of up to 4K through a pair of micro-HDMI ports. It has a capacity of up to 4 GB of RAM and a dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, and USB 3.0 options. Additionally, the Raspberry Pi 4 is equipped with Power over Ethernet (PoE) capabilities, made possible through a separate PoE HAT (Hardware Attached on Top) component.

Thus, the user can maneuver the ROV using a laptop computer and a gamepad controller. The interface is powered by the open-source QGroundControl application, which delivers a live video feed, sensor feedback, and critical information, alongside

the flexibility to modify settings and configurations.[31]
The connection between the ROV and the laptop is achieved by the Fathom ROV Tether, a 300 meters flexible polyurethane cable and suitable for marine applications, in Figure 3.4.

Figure 3.4: Fanthom ROV Tether. (Image credit: BlueRobotics)

The Fathom Tether offers a range of lengths, from 25 meters up to a maximum of 300 meters. For lengths beyond 50 meters, the utilization of the Fathom Spool is recommended to simplify management. The Fathom Spool is a water-resistant sturdy reel that streamlines the storage and handling of the tether connected to the BlueROV2. The tether boasts neutral buoyancy, a robust breaking strength between 300 to 350 pounds, and is infused with water-blocking fibers to prevent leakage. Embedded within the tether are one to four unshielded twisted pairs (UTP) of 26AWG wire. Connectivity is made with a pre-fitted Binder 770 plug, compatible with the FXTI and Fathom Spool. At the other end, a cable penetrator facilitates integration with the BlueROV2 or other watertight enclosures.[32]

SeaTrac Lightweight

The SeaTrac Lightweight is the underwater acoustic positioning system used in this project.
This system contains two beacons: the **X150** USBL beacon and the **X010** transponder beacon. This system is build upon the idea that one single USBL beacon can be used to track the positions of 1-14 underwater devices, each with its own transponder beacon, and install a bidirectional communication with them in real-time, up to a depth of 300 meters. This acoustic solution involves installation and operational constraints, but also the calibration of the USBL beacon. The system's set up requires that the X150 beacon must be powered up by the alimentation from a device on the surface, connected to a computer and drown into the sea, meanwhile the X010 beacon must be mounted on the underwater device. The X010 beacon returns

the distance between itself and the X150 beacon and this latter allows to know the remote beacon's relative position during the data exchange.

The X150 is the micro-USBL beacon, in Figure 3.5. It has a 9 DOF, AHRS and a Doppler sensor, which takes data from the onboard MEMS gyroscope, accelerometer and magnetometer to produce pitch, roll and yaw information.

The **accelerometer's** function is to calculate the pitch and roll orientations by discerning the direction of gravitational force in relation to the Beacons' local reference frame. It must record the raw sensor data when a force of +1G and -1G is exerted on each of its axes. These recordings are known as the limits. This calibration enables the device to interpolate measurements for forces between these limits and to disregard the acceleration that results from motion. These parameters are preset at the factory before shipment. Under normal circumstances, recalibration is unnecessary unless the beacons are subjected to extreme environmental temperatures.
The **magnetometer** measures the Earth's magnetic field lines and the direction of magnetic north in relation to the local frame of reference of the Beacons. It is crucial to note that the presence of ferrous materials, including Iron, Nickel, Cobalt, can influence the magnetometer readings when in close vicinity to the Beacon. Therefore, proper calibration of the magnetometer is essential before its deployment to guarantee that the yaw readings align accurately with magnetic north and are updated appropriately as the beacon undergoes rotation.
The **rotational rate-gyroscope** determines the rate at which the beacon rotates around each of its axes, relative to its local frame of reference. This gyroscope's readings are crucial for minimizing the response time of the Attitude and Heading Reference System and for facilitating swift updates to the yaw, pitch, and roll measurements.[33]

SeaTrac Beacons necessitate a DC power supply that is regulated and free from electrical noise. This power source must be capable of providing a steady voltage ranging from 9V to 28V, accommodating peak loads of up to 10W during acoustic transmissions.
The X150 uses a ASCII based command protocol. Data can be unpacked to get position and orientation information of the supervised underwater device relative to the Magnetic North and the gravity direction. [34]

Figure 3.5: X150 Beacon

The X010 beacon is the mini-transponder contained in the SeaTrac Lightweight system which returns the underwater device's data to the X150 beacon requests thanks to its internal DSP motor.[35] Each beacon is equipped with sensors that measure environmental pressure and temperature, enabling the calculation and continuous monitoring of the beacon's depth. Within a tracking system, this depth data from the remote beacons can be integrated into the position-solving process, enhancing the precision of vertical positioning. Furthermore, the pressure and temperature data contribute to the automatic refinement of the local velocity-of-sound value for each beacon, minimizing errors in ranging calculations. This integration ensures that the system maintains the highest level of accuracy in real-time operational conditions.

A navigation application, **SeaTrac NavPoint**, designed for Microsoft Windows is capable of displaying various parameters such as position, depth, attitude, and course history for each beacon within a user-friendly graphical interface. This application is engineered to handle a network of up to 14 remote beacons, all coordinated from a central USBL X150 beacon.

3.1.2 Surface Setup

GPS Receiver

The GPS receiver used in this project is a BU-353S4 model, Figure 3.6. It has a SiRF Star IV chip with a -163 dBm tracking sensibility, a fast TTFF to a low level signal and supports NMEA0183 protocol. It can be used indoor or outdoor, since it's water resistant, and it can be connected to a computer through its USB interface. It can be used to transform a certain device into a navigation device or in combination with software that require GPS input data.

Figure 3.6: GPS receiver BU-353S4

Laptop

The computer used is a Dell Latitude 7480 with a memory capacity of 15,5 GB and a disk capacity of 512,1 GB with an Intel® Core™ i5-7200U CPU @ 2.50GHz × 4 processor. In the computer is installed the Ubuntu 20.04.6 LTS Operating System of 64-bit. On the graphic side, it has a Mesa Intel® HD Graphics 620 (KBL GT2).

A Switcher (Figure 3.7) was used to create the communication between the Tether and the laptop, without using the FXTI box.

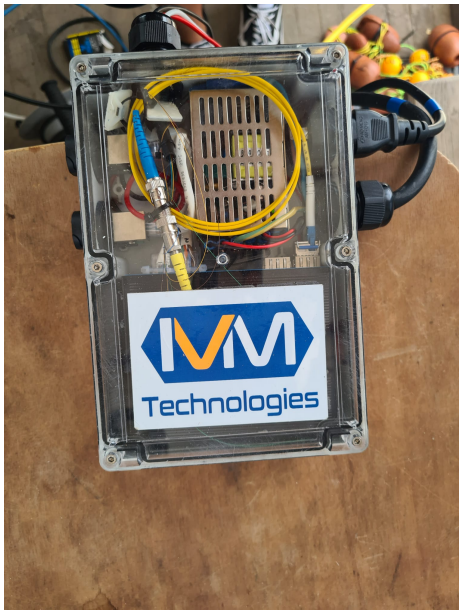


Figure 3.7: Switcher for ROV's communication

Outland Technology Power Supply

The OTPS system (Figure 3.8) is used to power up the BlueRov2 via tether instead of using a battery. The main priviledge in this is that there is no limited time of usage of the ROV.
The tether used for this purpose, receives a 100-240 VAC power, transforms it to a 400 V power, then transfers it through small wires and converts it again to 15 V until it reaches the ROV.

Figure 3.8: Outland Technology Power Supply

The OTPS contents involves a Topside Power Supply Unit, a ROV Power Supply Enclosure for OTPS and a High-Power Tether Cable with Connectors for OTPS of 125m or 250m.
The Topside Power Supply Unit is a compact suitcase which contains a built-in FXTI board and a security GFI which shuts down the system if any unsafe event or current leakage is detected. When it is opened, it shows a lot of features such as: a ground wire connection, a tether connection in which will be connected ROV's tether and a power cable input to power it up. It also has an outlet for the computer charger, a power switch to let the current flow into the topside suitcase. On the lower part of the topside there is a display which shows the status of the ROV in real-time and a GFI test switch. The middle part is designed to be the laptop tray. On the right side there is a user access panel, a USB-B and 124 an external FXTI connection.
The ROV Power Supply Enclosure is mounted on the lower part of the ROV and connected to the tether in the ending part, where it converts the 400 V power into the 15 V.

The High-Power Tether Cable with Connectors is a particular tether designed with five twisted pairs where two of them are used for an isolated power transmission, another couple is used for an earth ground connection and the last one is used for the Fathom-X communication.[36]

3.1.3 Software Applications

ArduSub

ArduSub (logo in Figure 3.9) is a fully-featured open-source software for ROV. It is a part of ArduPilot project and acquire features from the ArduCopter code, such as simulators, log analysis tools, and advanced APIs for comprehensive vehicle management and control. It provides features like feedback stability control, depth and heading maintenance, and the facility for autonomous navigation. ArduSub integrates with Ground Control Station software, enabling users to monitor vehicle telemetry and engage in sophisticated mission planning activities. ArduSub firmware is compatible with Pixhawk, Pixhawk 2 and Pixhawk Mini platforms.[37]

Figure 3.9: ArduSub logo. (Image credit: BlueRobotics)

MAVLink

MAVLink (logo in Figure 3.10) is a lightweight messaging protocol for sending and receiving messages between aerial drones or underwater vehicles and other components, usually ground stations. MAVLink uses a pattern that combines elements of both publish-subscribe and point-to-point approaches: information streams are transmitted as topics, while configuration sub-protocols employ a point-to-point model with retransmission. XML files are used to define supported MAVLink messages, which constitute the dialect. Especially the low level and general purpose Pymavlink library was used in this project to read data from sensors and send commands through a Python script. [38]

Figure 3.10: MAVLink logo. (Image credit: mavlink.io)

QGroundControl

QGroundControl (logo in Figure 3.11) is an open-source, cross-platform ground control station software, created for autonomous or remotely controlled vehicles. It serves as a comprehensive tool for planning, monitoring, and controlling UAV missions of any vehicle which supports MAVLink communication. It offers features such as mission planning, waypoint navigation, live video streaming from onboard cameras, manual control of the vehicle and real-time telemetry data display, including vehicle position, altitude, battery status, and sensor readings. QGC runs on Windows, OS X, Linux platforms, iOS and Android devices.
The QGC 4.2.6 version was used in this project. [39]

Figure 3.11: QGC logo. (Image credit: qgroundcontrol.com)

ROS Noetic

ROS stands for Robot Operating System and it is an open-source collection of libraries and tools for robotic applications' developers and users. This can be used in multiple domains, such as indoor robots or automotive and in multiple platforms, such as: Linux, Windows and macOS. Its main usages are for robots' autonomy implementation, backend management and user interfaces, but there are more topics that can be explored with ROS. Basically, it can operate on robots' actuators, sensors and control systems and connect them together. Each ROS process can be represented with a node in a graph. The nodes can send, receive and split messages from or to other nodes, actuators or sensors. Moreover, ROS' features are indipendent to the programming languages that can be used (C++, Python and LISP). In this project the ROV Noetic version (logo in Figure 3.12 is used since it is compatible and designed work with the Ubuntu 20.04 OS. [40]

Figure 3.12: Ros Noetic logo. (Image credit: ROS.org)

Unity3D

Unity is a real-time 3D cross-platform game engine and development framework used to create interactive 3D, 2D, virtual reality (VR), and augmented reality (AR) applications. It provides a comprehensive set of tools and functionalities that enable developers to design, develop, and deploy games, simulations, and other interactive experiences across various platforms, including desktop computers, mobile devices, gaming consoles, and even web browsers.
It is available for Windows, macOS and Linux. It supports scripting in C (C Sharp) and provides a powerful scripting API.
The version used for this project is Unity 2021.3.18f1.[41]

Figure 3.13: Unity logo. (Image credit: unity.com)

3.1.4 Final Architecture

The final composition consists of a Dell Latitude 7480 laptop positioned on the Topside Power Supply Unit powered up and connected to the BlueROV's tether. The X010 beacon of the Seatrac Lightweight must be mounted on the BlueROV2, meanwhile the X150 beacon must be attached to one of the laptop's USB port, then

both can be inserted in the water.
Meanwhile, the software bridge was created towards a ROS file and launching both the python script and the QGroundControl app at the same time. The QGroundControl app was previously set up to allow bidirectional communication between the script and the ROV.
The default communication port of QGroundControl is the 14550 port, but it was changed to 14440 to allow ROS data to be sent to the 14550 port.
The communication between the ROV and the script is created through the PyMavlink library which allows to receive and send data in a bidirectional way.
ROS libraries are used in the script to create topics in which messages can be both written or read. These topics open the communication with the Unity project even though the C# programming language is used instead of Python.
The local NED coordinates calculated in the Python script thanks to the Seatrac Lightweight measurements and the ROV's data are transferred to the C# script in Unity to update the ROV digital-twin position in the enviroment. To create the connection between the C# script and the ROV digital-twin the properties of the latter are modified and the script is linked to the 3D ROV.

3.2 Development of a LOS control system

This section explains the developed target tracking controller which allows the autonomous navigation of the BlueROV2 between a predefined starting point and a destination in terms of latitude and longitude. This is achieved by guiding the ROV along a straight trajectory based on a Line of Sight (LOS) control strategy, which involves a sequence of movements: initially vertical, followed by rotation around the yaw axis, and finally, forward motion.
Thus, the section introduces the Line of Sight control strategy and its practical software implementation.
Especially, it provides a solution of the positioning challenge, using the SeaTrac acoustic sensor, through the implementation of a message exchange between the device and the ROV, and the a real-time control strategy based on the information received.

3.2.1 Line of Sight Guidance

Line of Sight (LOS) guidance is a strategic approach in navigation categorized as a three-point guidance scheme for underactuated vehicles. The term "line of sight" is derived from the tactical requirement that the interceptor's movement is directed along the LOS vector, which connects the reference point and the target, as shown in figure 3.14. [28]

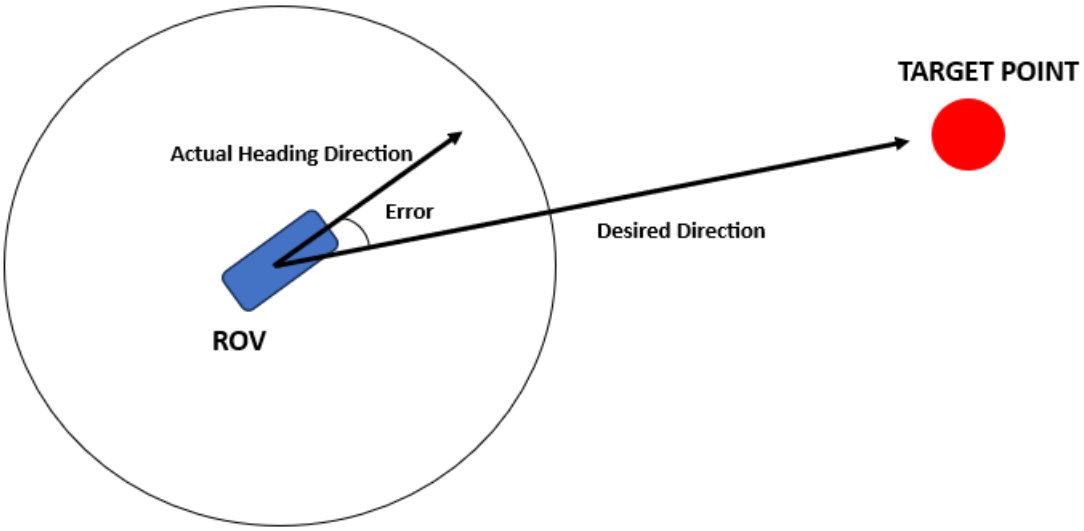


Figure 3.14: Heading LOS guidance strategy

This approach transforms the path following control problem into a heading control problem, by aligning the vehicle's movement with a dynamically calculated reference point along the intended path. This alignment is achieved through constant adjustments to the vehicle's heading, ensuring that it remains on course towards the next waypoint, in order to simplify the navigation process.[42]
Thus, the strategy is particularly valuable in applications where precise and straight line movement is essential, like surveying, mapping, and targeted exploration. An application example of this strategy is given by [43].

3.2.2 Implementation of the NGC system

The aim of the developed algorithms is to render the BlueROV2 autonomous in navigating from a known starting point to a designated destination, defined in terms of latitude and longitude, using a Line of Sight guidance approach. The ArduSub firmware offers intrinsic control strategies for vertical movement and rotation around the yaw axis.
In the following sections, the configuration of the system and the underlying logic of the algorithms are explained.

Configuration

The configuration process of the navigation system involves several steps.
Initially, the X150 beacon must be calibrated using the SeaTrac Tools application on a Windows platform. The X010 beacon has to be mounted on the structure of the BlueROV2 and the ROV's tether has to be attached to the Topside Power Supply Unit, which is equipped with a tether plug-in. The system setup includes also placing a computer on the Topside Power Supply Unit, powering it up with a

cable connected to the Topside's board.

For the purpose of data acquisition and positioning, the GPS Receiver has to be connected to a USB port on the computer, and similarly, the X150 beacon to another USB port. The GPS Receiver has to be placed on a flat surface, and the X150 beacon should be inserted into the water directly below it, ensuring they share the same latitude and longitude coordinates and paying attention to be at least 1 meter away from all surfaces. This placement prevents the creation of multi-path signals caused by sound waves reflecting off surfaces, which could lead to implausible beacon readings. Moreover, it is important to maintain a distance of more than one meter between the BlueROV2 and the X150 beacon to avoid any signal interference and ensure accurate positioning and navigation. Additionally, the Ethernet cable from the IVM power box is connected to an Ethernet port on the computer, and the IVM power box is powered up using its power cable.

Thus, the system is ready for deployment with the insertion of the ROV into the water and the activation of the Topside Power Supply Unit via its switch. Upon successful powering up, audible and visual indications confirm the operational status, such as a sound emanates from the ROV, lights inside the ROV's cylinder illuminate, and the LED on the X150 beacon begins flashing green.

The schematic of the system configuration is shown in Figure 3.15.

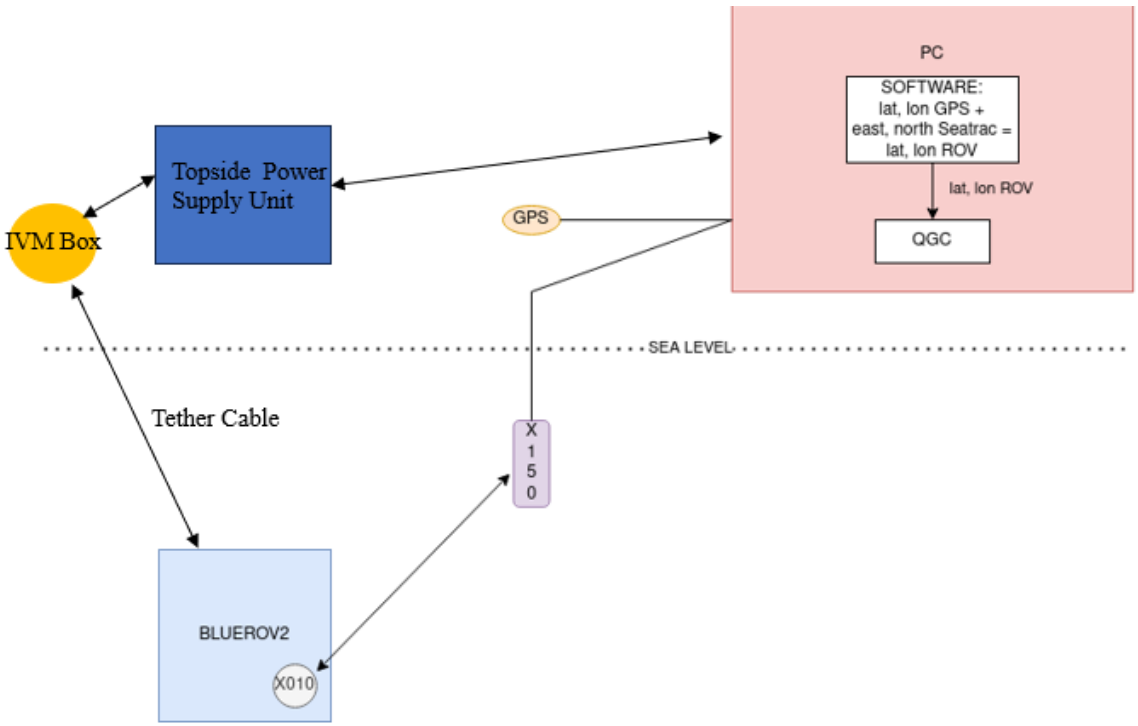


Figure 3.15: Complete configuration

Regarding the software configuration Ubuntu 20.04 was installed as operating system, Visual Studio Code version 1.80.1, Python version 3.8.10 and QGroundControl version 4.2.6.

This last software has to be configured in order to ensure seamless communication and interaction with the ROV. By default, QGroundControl uses port 14550 to communicate with the ROV when the application is not open. To facilitate communication through a Python script when the application is active, a new communication port, 14440, was set up. This necessitated configuring QGroundControl to forward information received from the ROV to this new port by enabling all boxes in the *"Ground Station"* section of the MAVLink settings and changing the *"Host Name"* to localhost:14440, as shown in Figure 3.16.

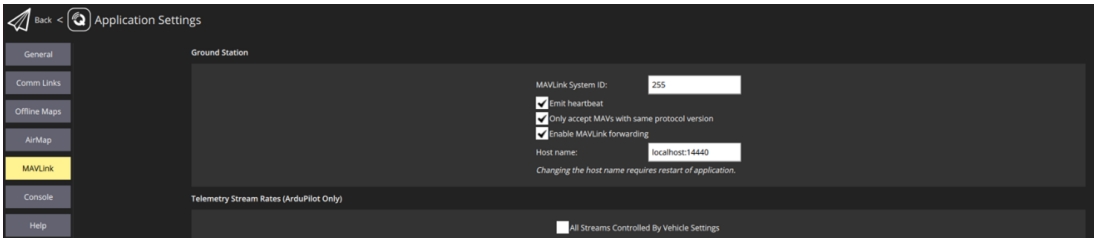


Figure 3.16: QGC local port configuration

Another adjustment in QGroundControl involved addressing a potential conflict with the default auto-connection feature. QGroundControl automatically connects to all devices attached via USB, which could interfere with the Python script's access to these devices. To resolve this, the auto-connect feature was disabled, allowing the Python script to communicate with the USB connected devices without conflicts by disabling all boxes under *"AutoConnect to the following devices"* in the *"General"* section of the application settings, as shown in Figure 3.17.

Figure 3.17: QGC auto-connection configuration

Algorithms

The software wad developed in Python and it allows message transmission between QGC and the ArduSub firmware on the Raspberry Pi4 of the ROV, through PyMAVLink protocol, acting as an intermediary.

Two codes have been developed: the first dedicated to the real-time localization of the ROV, and the second utilizes this positional data to execute a control strategy targeting a specific point.

Especially, the software is responsible for a variety of tasks:

- **Opening of serial communications**: The code manages the serial communication between the laptop and the X150 and the GPS. It also provides a communication between the software and QGC, through *mavlink_connection()*, allowing for listening to or sending messages to port 14440 and waiting for a *"heartbeat"* before executing any command.
- **Message Reading and Decoding**: It requests status messages from the X010 beacon, which are then received by the X150 beacon, establishing a reliable communication link between the beacons. It reads and decodes all status messages sent to the X150 beacon from the X010 to extract ROV's NED coordinates relative to the X150 beacon.
- **GPS Integration**: It reads the position of the GPS receiver in latitude and longitude (so the position of X150 beacon) and utilizes these coordinates to calculate the ROV's latitude and longitude, through Haversine formula.[1]
- **Real-time updating QGC**: The script sends the ROV coordinates to QGroundControl, enabling the display of its position on a real-time map.
- **Managing the control system**: The script sends a variety of commands to the ROV's motors. These commands are designed to achieve the desired depth, maintain the correct orientation, and enable the ROV to reach the predetermined endpoint.

The decision was made to implement a threaded logic framework to facilitate concurrent processing, requiring the simultaneous execution of different tasks.
A total of six threads are utilized in the system, in the sequence of execution:

1. **QGround Loop**: This function manages the interaction with QGroundControl, continuously reading MAVLink messages, through *master.recv()* and sending the coordinates of the ROV, to QGroundControl via the *master.mav.gps_input_send()* function, through ArduSub.
2. **Reading Loop**: The reading function reads and interprets messages from the Seatrac system, decoded from bytes into ASCII. When a *CID_NAV_QUERY_RESP* message is received, which is a response to the Writing Loop's message, the Decoding function is triggered. It decodes the NED coordinates and orientation of the ROV.

[1] The haversine formula provides an extremely precise method, especially for smaller angles and distances, for calculating the distance between two points on a sphere's surface, based on their latitude and longitude. Further information about this formula can be found in [44]

3. **Writing Loop**: It sends periodic requests to X150 beacon for updated data, every 4 seconds, through *CID_NAV_QUERY_SEND*.

4. **Locating Loop**: The locating function processes GPS data to pinpoint the ROV's latitude and longitude, when a *GPGGA*-type messages is received. Then it calculates the latitude-longitude coordinates of the ROV by converting the NED data with the inverse of Haversine formula and by adding the GPS information.

5. **Log Loop**: This function sets up the logging files and writes the ROV's current state (position and orientation).

6. **Control Loop**: It is the central part of the control system. The function starts by arming thrusters and setting the ROV to *ALT_HOLD* mode to maintain a specific depth.
   Then the loop continuously monitors the ROV's current latitude, longitude, and depth. It compares these values against desired coordinates, which represent the target location and depth for the ROV.
   It calculates the difference between the ROV's current and desired states. If there's a difference in depth, unless within a certain error, it sends commands to adjust the ROV's motors to achieve the desired depth, ensuring vertical positioning accuracy.
   If the desired depth is reached, it determines the bearing needed to reach the target point from the actual position and it adjusts the ROV's orientation to align with this calculated yaw angle.
   Once the depth and yaw are aligned with the desired values, the loop checks the distance to the target location. If the ROV is not within a specified range of the target, it commands the ROV to move forward, guiding it towards the target point.When the ROV is within the target range, the loop commands to stop the forward movement, indicating that the ROV has reached its intended destination.

In Figure 3.18, a flowchart diagram of the software is shown.

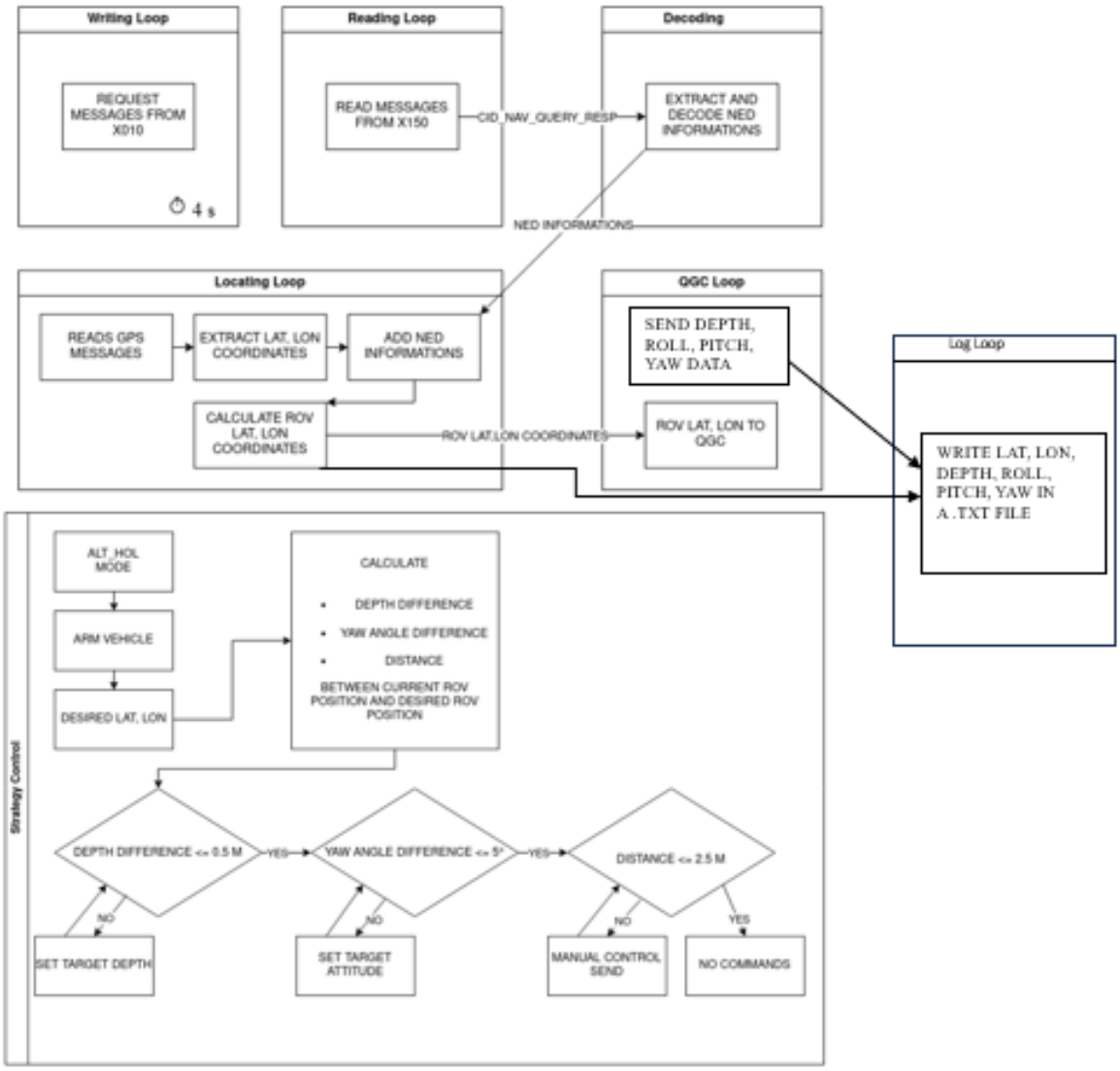


Figure 3.18: Flowchart diagram of the software

3.3 Development of an experimental Digital Twin environment

The second part of this study consisted of the integration of a project developed by Eliott Deleplanque for his thesis work, with the algorithms formulated in the preceding section.
Deleplanque's project used a simulation conducted in ArduSub to gather data from the simulated ROV, and then transmitting this data to a digital twin in Unity to mimic the behavior observed in the simulation.
Modifications were made to Deleplanque's code to enable its functioning in real-time rather than solely in a simulated environment. This revised code processes incoming

messages from the ROV, Seatrac, and GPS. The collected data are then processed and synthesized to calculate the ROV's current position in the water, which is subsequently displayed on QGC in real-time. The ROV's position is converted into NED coordinates, which are then relayed to the digital twin in Unity to replicate the behavior of the actual ROV.
To achieve this, a connection was initially established between the script and the ROV for data acquisition, followed by establishing a link between the script and the ROS for sending commands to the digital twin in Unity.
Unfortunately, this system has not been tested due to the malfunction of the ROV.

3.3.1 Unity Digital Twin

For this project, an accurate digital model created by Deleplanque in Unity was employed, depicting the system comprising the ROV, in Figure 3.21, and the dock of the Port Point Rouge of Marseille, in Figure 3.19. To this model, a reconstruction of the seabed, crafted by Pierre Drap, was added, with the points of interest (red squares), shown in Figure 3.20. This integration provided a comprehensive and realistic representation of the underwater environment, where tests were conducted.

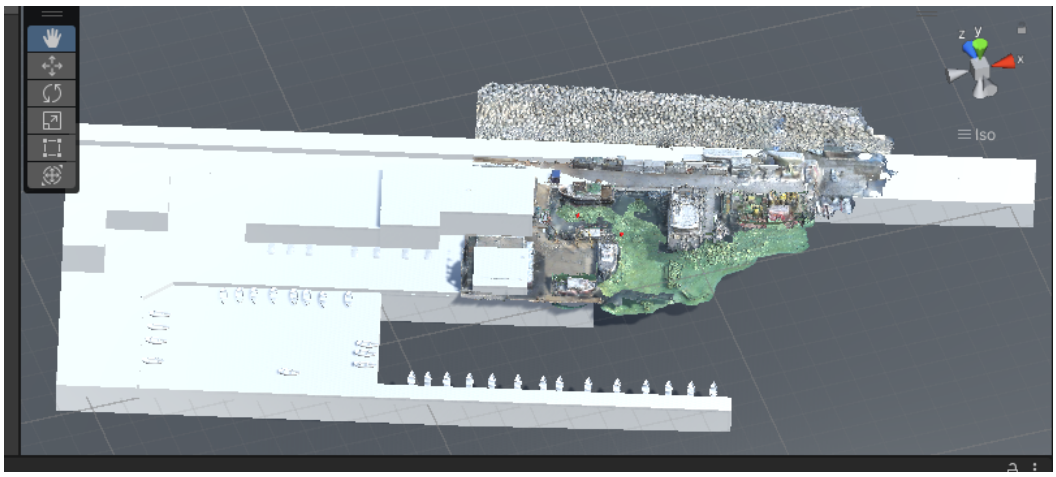


Figure 3.19: Digital representation of the dock in Unity

The simulation in Unity focuses on replicating the behavior of the BlueROV2. This is achieved through dedicated scripts that establish a connection to the ROS topic. The primary purpose of these scripts is to wait for and receive messages containing information about the movements and orientation of the ROV. Once received, these messages are converted into a format (Vector3) compatible with Unity, and orientation is extracted as a Quaternion. With the data received, the simulation updates the matrices that manage the position and orientation within the Unity environment. This allows for an accurate reflection of the real-world movements and orientation changes of the ROV in the virtual context. Essentially, every movement or change in direction of the ROV in the real world is translated and replicated in the simulation,

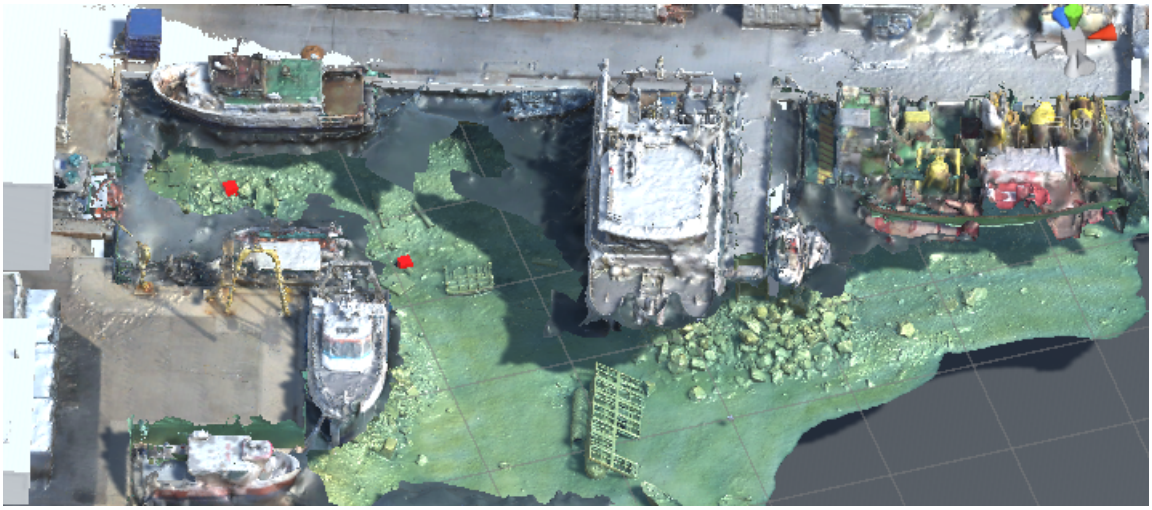


Figure 3.20: Digital representation of the seabed in Unity

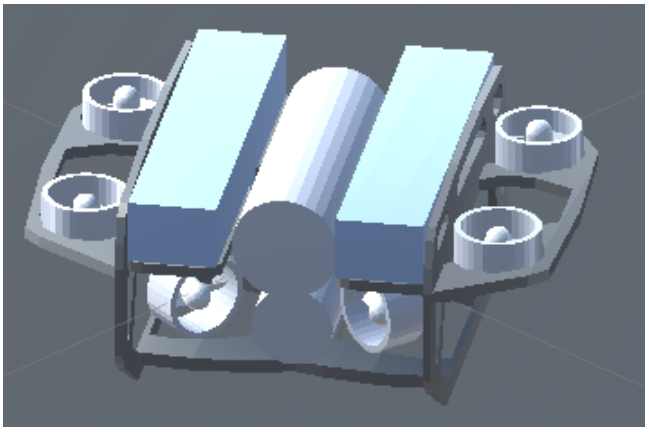


Figure 3.21: Digital Twin of the ROV

providing a realistic and consistent experience with the operations of the ROV.

3.3.2 Implementation of the NGC system

The main aim of the control system is to solve a path following problem, reaching desired waypoints, to scan the seabed and find some objects. The logic behind this control system is the same of the first part, using the Line of Sight approach.
In the following sections, the configuration of the system and the underlying logic of the algorithms are explained.

Configuration

The hardware configuration employed in this part of the study mirrors the setup delineated in the preceding section, therefore, for the sake of simplicity, it is not reiterated here.
However, the configuration scheme is shown in Figure 3.22. Regarding the software

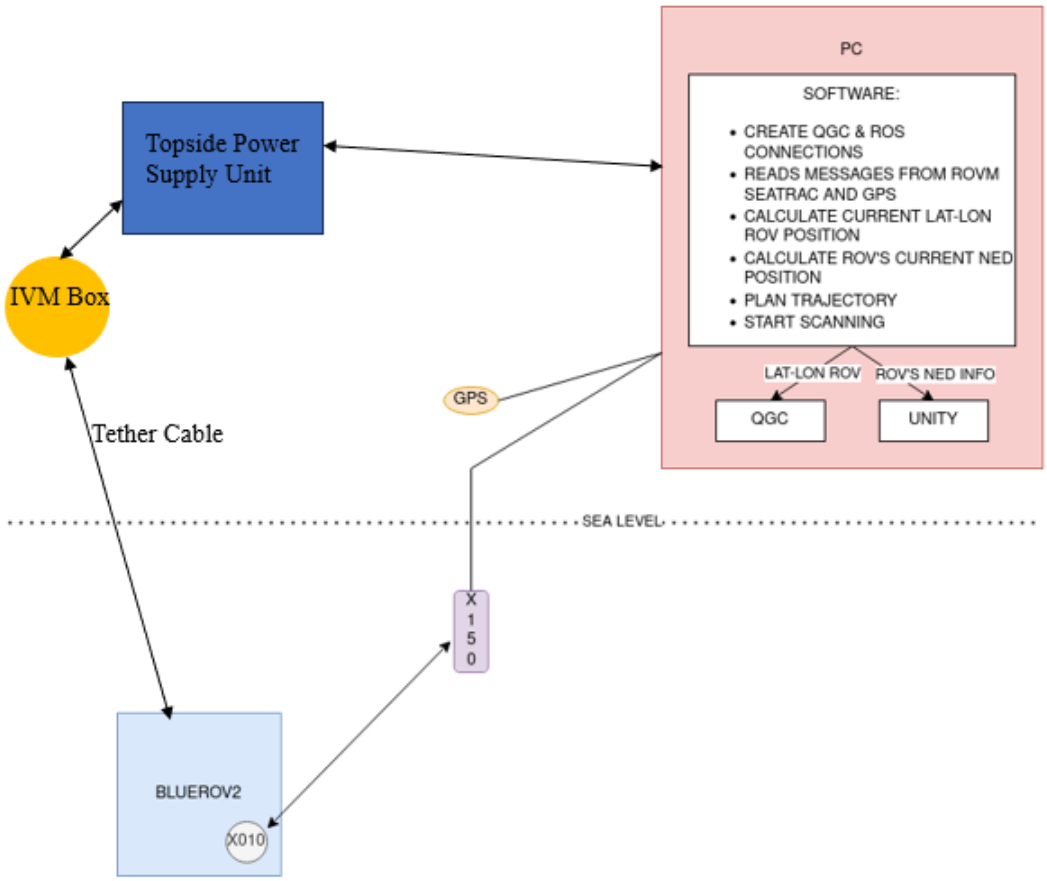


Figure 3.22: Complete Configuration

setup Ubuntu, Visual Studio Code, Python and QGroundControl was installed with the same versions and configuration done in the first part.
It was also installed ROS Noetic, compatible with Ubuntu 20.04, and Unity 2021.3.18f1. For further details about the installation and the configuration of the simulated environments, readers are directed to consult Deleplanque's repository `https://github.com/ELTGR/Mission_Planning`.

The software communication system between the ROV, ROS and Unity, shown in Figure 3.23, can be analyzed into three parts: ROV-Python communication, Pyhon-ROS communication and ROS-Unity Communication. To communicate with the

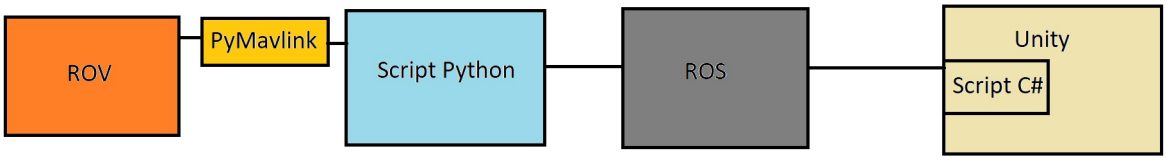


Figure 3.23: ROV-Python-ROS-Unity communication

ROV, PyMavlink library is used, providing functions to send and receive data.

ROS facilitates communication between programs that may be written in different programming languages. In the context of this study, it is utilized to transfer data from a Python script to a C# script, thereby updating the position of the ROV in Unity. A topic named *"/local_position"* is defined. This topic is set up for publishing, meaning that it is designed to write data. Thus, It receives the position data of the ROV from PyMavlink and publishes this data to the previously defined topic.

To update the position of the ROV within the Unity environment, a C# script is utilized. This script is located in the Asset directory of the Unity project, linking the script with the ROV model and allowing for real-time updates of the ROV's position in the Unity simulation by reading data from the topic.

Algorithms

The software is responsible of:

- **Communication and Connection Setup**: It establishes a connection between the Python script and ROS, facilitating communication with the Unity Project through the 14550 UDP port. It also creates a link between the Python script and QGroundControl to communicate with the ROV via the 14440 UDP port.
- **Message Reading and Decoding**: The code requests and receives status messages from the X010 beacon via the X150 beacon and decodes them to determine its NED coordinates.
- **GPS Integration**: It reads the GPS receiver's position in latitude and longitude, using these coordinates to calculate the ROV's underwater position, through Haversine formula.
- **Real-time updating QGC**: It sends the ROV's coordinates to QGroundControl for real-time mapping.
- **Trajectory Planning**: It plans and creates the entire trajectory of the ROV. Initiates scanning operations, storing data in .txt files, and sends the NED coordinates to the Unity Project to mimic the real-time movements of the ROV.
- **Managing the control system**: The software adjusts the ROV to a specified depth and calculates its current NED coordinates relative to the desired lat-lon position. It transforms lat-lon points into NED coordinates for precise navigation and sends commands to thrusters to follow the desired path.

The software is composed of various scripts, each dedicated to manage different tasks:

- **bluerov_node.py**: It establishes communication with ROS and handles the creation, sending, and publishing of ROS messages. This script processes information from the ROV, Seatrac, and GPS, allowing it to calculate the current position of the ROV. This data is then sent to QGC for real-time display of the vehicle's location on a map. Moreover, these details are converted into NED coordinates, crucial for initiating the scanning process. Essentially, bluerov_node.py enables the BlueROV2 to autonomously navigate between a known starting point and multiple predetermined points of interest.
- **bridge.py**: It enables the transmission of commands to its motors and the retrieval of data from connected devices such as Seatrac and GPS. This script also incorporates functionality for scanning and maneuvering the ROV towards specified NED coordinates.
- **Config_scan.yaml**: The file stores details for scanning operations.
- **Gridy_based.py**: The script is responsible for outlining the trajectory that the ROV follows, ensuring a structured approach to navigation.
- **return_on_target.py**: If the ROV detects objects, this script contains functions that enable to return to the points where these detections occurred.
- **visualisation_rov_way.py**: It designs the ROV's trajectory using data logged in log_position.txt file. This file records the ROV's state (time, x, y, z, yaw) at each update. Other files are stored such as the sonar_time.txt, that logs data when objects are detected by the sonar, or the start_stop_recording.txt, that marks the start and stop points at each waypoint reached.

The software described is partially derived from Deleplanque's work, yet it has been significantly adapted to the NGC developed in the previous section.
The main contributions are: including global variables for thread management, message reading from QGC, Seatrac, and GPS, managing of the ROV's lat-lon and NED coordinates, roll, pitch, and yaw and decoding of Seatrac's stream data through functions for sign value extraction from bit data or hex to decimal conversion. It also implements a function for calculating the ROV's current NED coordinates relative to its initial lat-lon coordinates and facilitates ROV communication via the 14440 UDP port, including functions for interacting with QGC. Moreover it sets up the 14550 UDP port for ROS communication, calculates lat_ref, lon_ref from GPS and Seatrac NED data and handles East, North, Down variables. It manages various operational loops like Reading Loop, Writing Loop, Decoding Loop, NED Loop, and QGC Loop.
In Figure 3.24, a flowchart diagram of the software is shown.

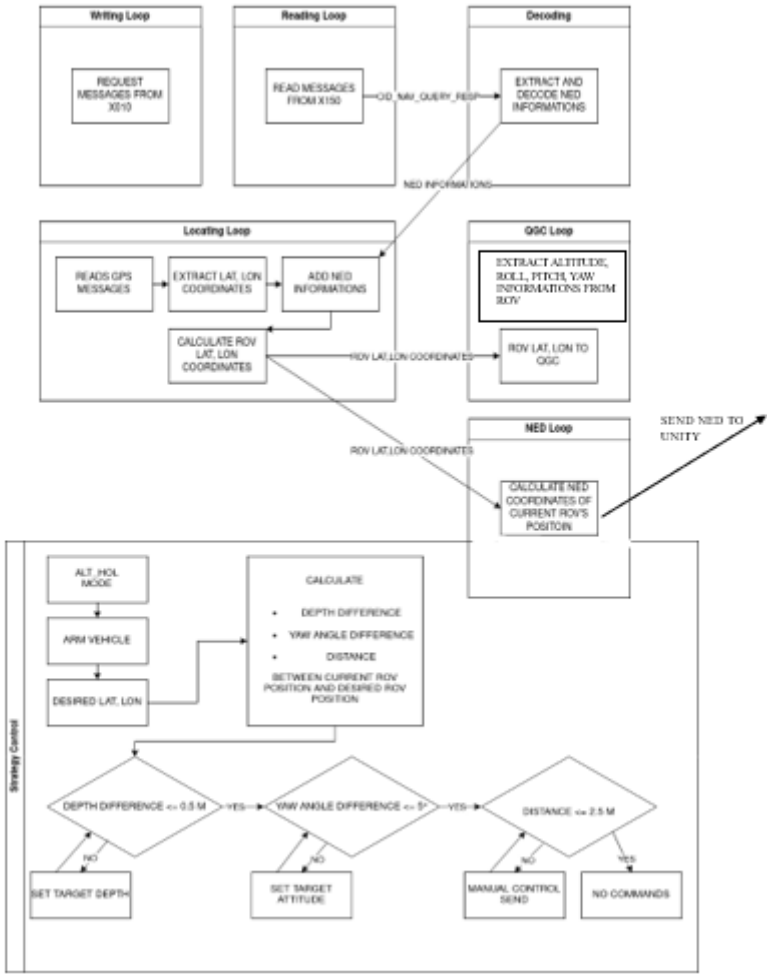


Figure 3.24: Flowchart diagram of the second software

Chapter 4

Experimental results

This chapter presents the performance of the NGC software, evaluating the efficacy under sea conditions, including navigation precision, response to control inputs, stability under varying sea states, and data communication efficiency.
These trials were conducted in two distinct aquatic environments: a controlled setting within the "Lycée Marseilleveyre" swimming pool and the natural environment of the Marseille Pointe Rouge port.
This chapter particularly focuses on presenting results collected during field tests conducted at the port, where both the control system and localization capabilities of the software were rigorously evaluated. Notably, the comprehensive version of the software, which includes the integration of a digital twin, was not tested in these field trials. Therefore, the second section of this chapter offers insights into the performance of this integrated system through simulation-based results.

4.1 Tests of the NGC system

After the hardware and software was correctly configured, the BlueROV2, with the X010 acoustic beacon, was submerged alongside the X150 beacon, checking if no other acoustic devices were present and if the vehicle was properly balanced, shown in Figure 4.1.

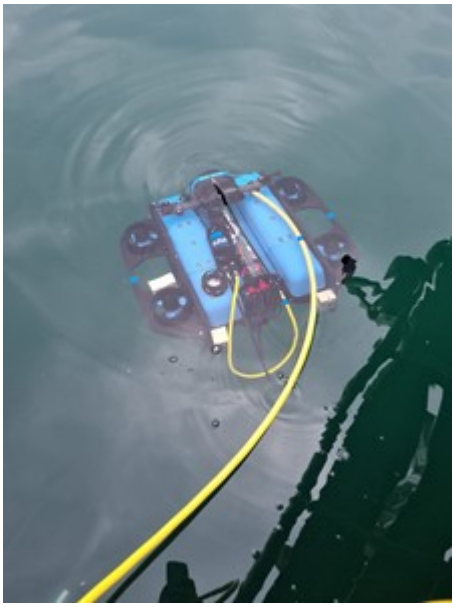


Figure 4.1: ROV's Buoyancy

Thus, the Python script for only localization was run via Visual Studio Code, followed by opening the QGroundControl App, to check if the ROV was correctly positioned on the map and if the devices were communicating properly. Then, thrusters were tested by using a joystick. Once their functionality was confirmed, the NGC software was tested, collecting data about time, Latitude, Longitude, Depth, Roll, Pitch and Yaw.
In this way, the ROV, from its initial point, reached the desired target by adjusting its depth, its yaw angle and moving forward. The desired longitude and latitude of the target and the GPS position, used for this results, are shown in Figure 4.2. Then,

```
Lat_Des------------------Lon_Des------------------Depth_Des------------------GPS Lat------------------GPS Lon
43.24382333333333,5.363253333333334,2.5,43.24382333333333,5.363253333333334
```

Figure 4.2: Desired latitude and longitude and GPS position

a series of graphs were created to evaluate the control system. The graphs were generated through MATLAB scripts, specifically designed to import and process data collected during the experiments.
Initially, data was cleaned, removing rows where the reading of messages from the Seatrac had not yet occurred. It is also important to note that the data collection occurred at four-second intervals. The graphs are explained as follow:

- **3D Trajectory Graph**: Using latitude, longitude, and depth, a trajectory graph of the ROV was created, as in Figure 4.3. The vertical axis represents depth, which appears to fluctuate throughout the course of the trajectory. This suggests that the ROV is responding to control commands aimed at reaching its depth, conditioned by the disturbances in the water. Additionally, it can

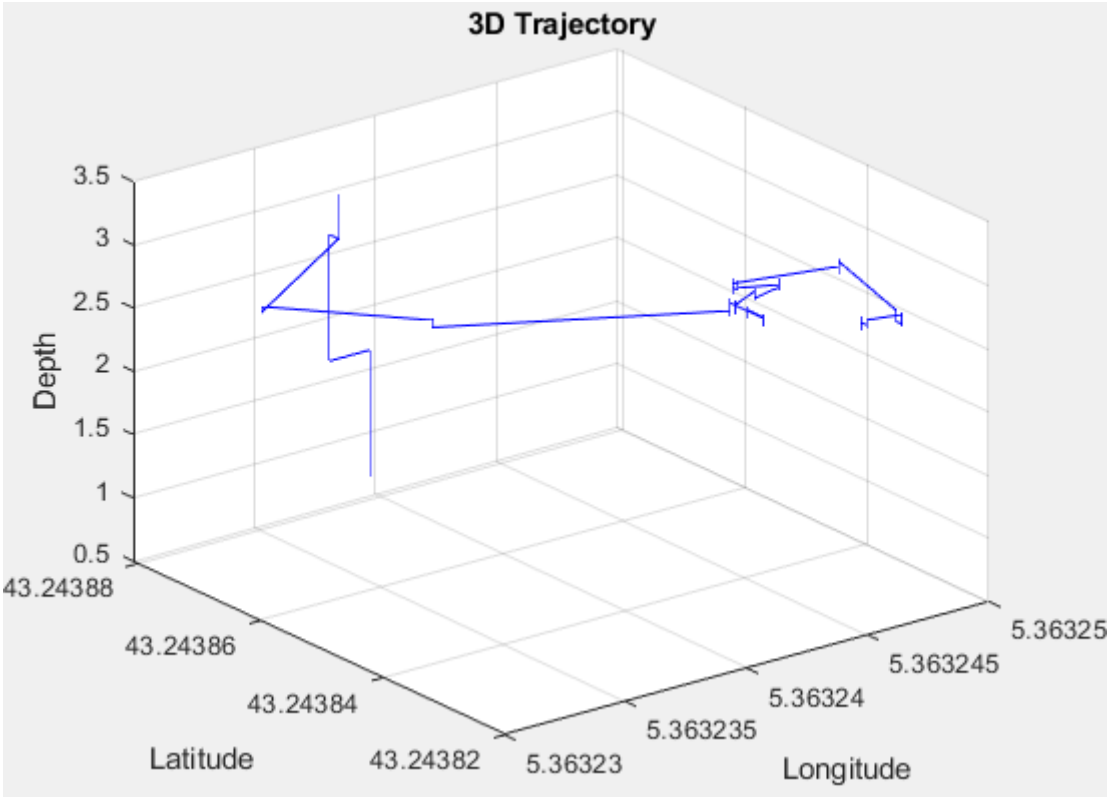


Figure 4.3: 3D trajectory graph (lat-lon-depth)

be observed that the ROV first adjusts its depth and then initiates movement towards the target.
The trajectory in the plane formed by latitude and longitude shows the path taken by the ROV. It should be a linear path, as a LOS strategy was used, but external deviations, like current, may affect the trajectory. Anyway the control strategy can overcome these disturbances. The ending point of the trajectory, compared to the desired final location, will indicate the NGC system's accuracy in reaching target coordinates.

- **Time-Depth Graph**: The graph in Figure 4.4 shows how the depth of the ROV varies over time, useful for analyzing the dynamics of the system. This graph underline a short transient response, indicating how quickly the control system is able to react to the command and start the descent process. After the initial descent, the depth line becomes relatively horizontal, with minor fluctuations around the desired depth value[1]. This indicates that the ROV has reached the desired depth and is maintaining it, which could be reflective of the 'hold depth' capability of the control system. Once again, the slight variations along the horizontal part of the line suggest some oscillation around the target depth, which might be caused by the ROV's buoyancy control adjustments or

[1]The desired depth value is -2.5 meters but there is a tolerance of ± 0.5 meters

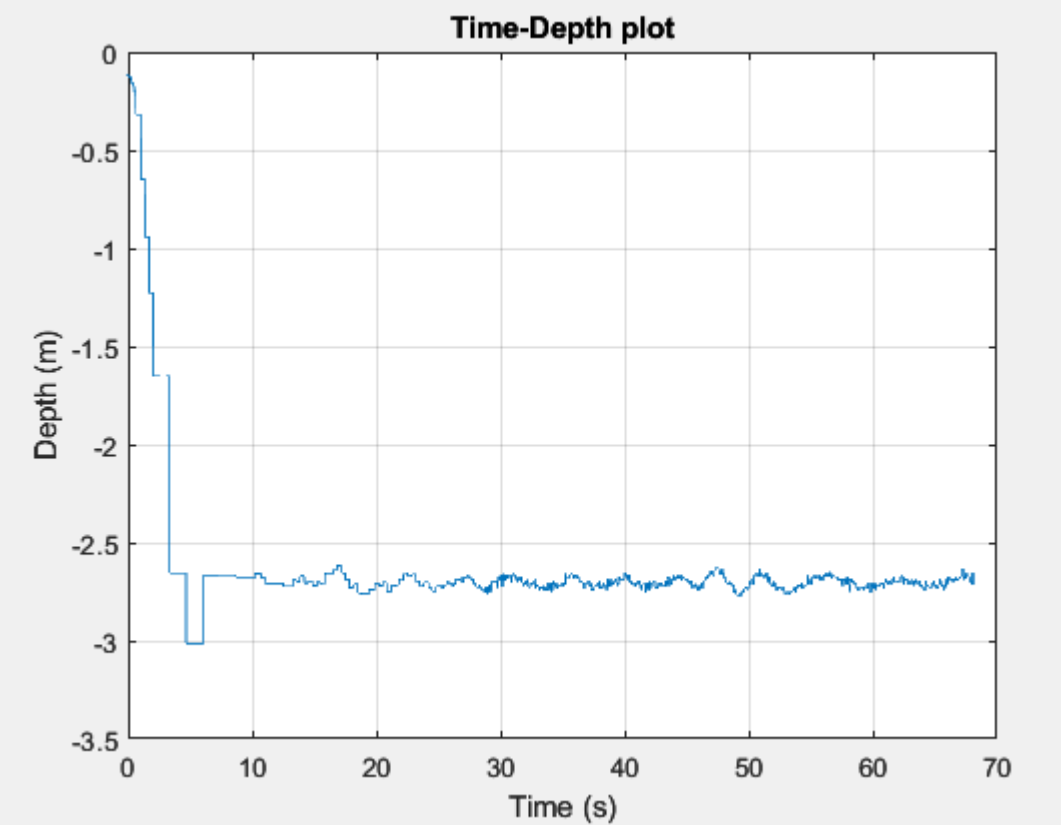


Figure 4.4: Time - Depth graph

environmental factors.

- **Orientation Graphs (Roll, Pitch, Yaw)**: Separate graphs for roll, pitch, and yaw over time, in Figure 4.5 analyze the Rov's balance and orientation control. The roll graph fluctuates around a relatively stable mean, indicating the ROV's side-to-side tilting is minimal. Except for a significant initial fluctuation, attributed to a slight imbalance in the motor armature, the graph suggest a system actively maintaining equilibrium.
  The pitch graph shows more significant deviations than the roll, suggesting an imbalance for the front-to-back orientation.
  The yaw graph demonstrates adjustments made to align with the target heading.

- **Latitude and Longitude Graph**: To observe how the ROV moves geographically, latitude and longitude over time was plotted, shown in Figure 4.6. The stepped pattern is due to the fact that the latitude and longitude data are updated every 4 seconds. Moreover, the pattern of movement suggests that the ROV is stopping or slowing at intervals, which may align with the heading or depth adjustments in the navigation path.

- **Speed Analysis**: The speed was calculated, based on changes in latitude, longitude, and time, in Figure 4.7. In this graph is more evident how the 4

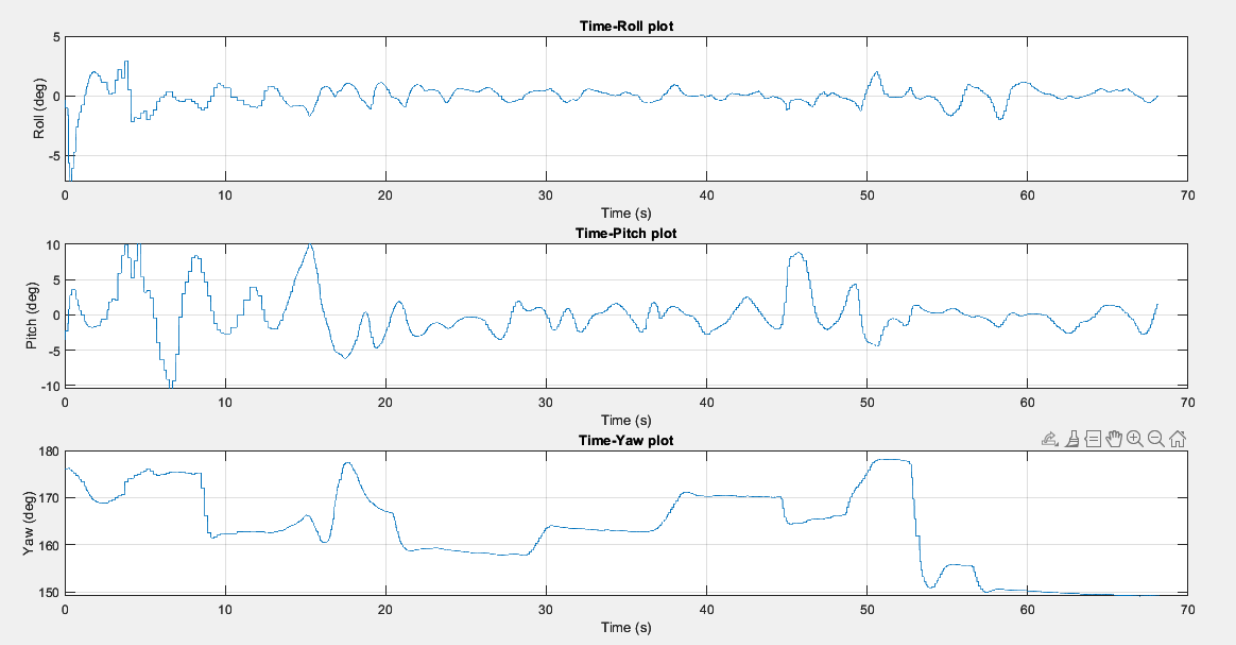


Figure 4.5: Roll, Pitch, Yaw graphs

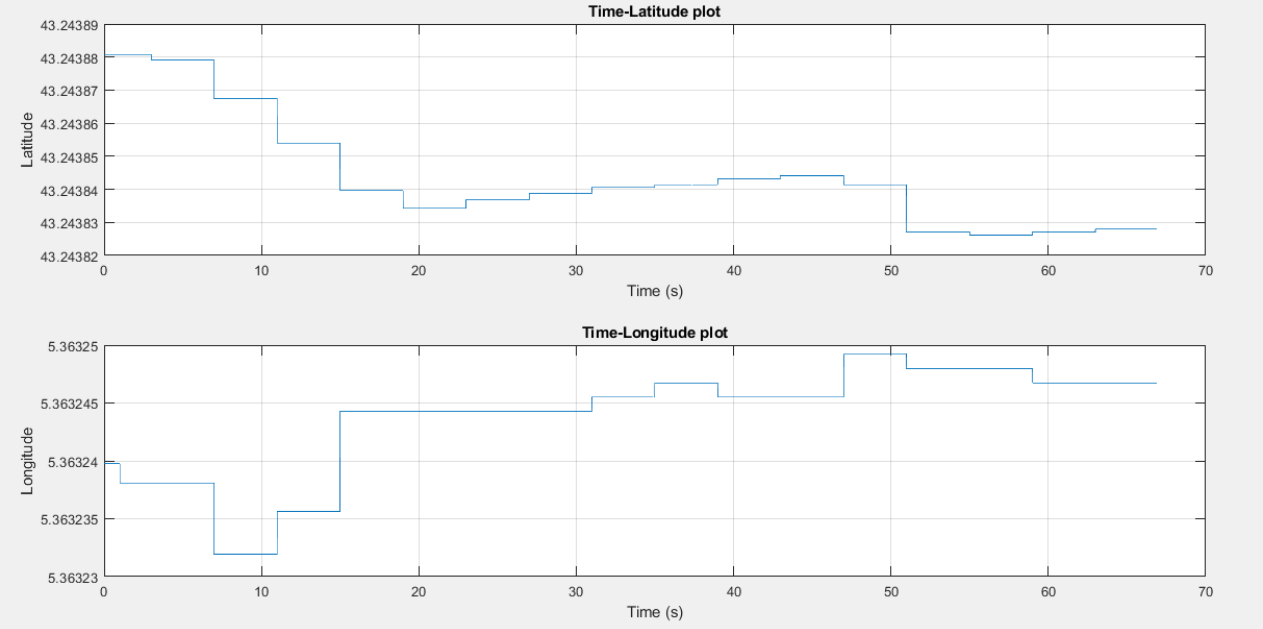


Figure 4.6: Time - Lat,Lon Graph

seconds data updating causes snapshots at these intervals. Hence, the graph is not a smooth speed curve but rather a series of discrete readings that capture the speed at specific moments. Furthermore, comparing this graph with those of the trajectory and yaw, it is observed that a yaw adjustment occurs around 50 seconds, possibly due to increased interference from the external environment, leading to a slight deviation from the reference trajectory and, consequently, an

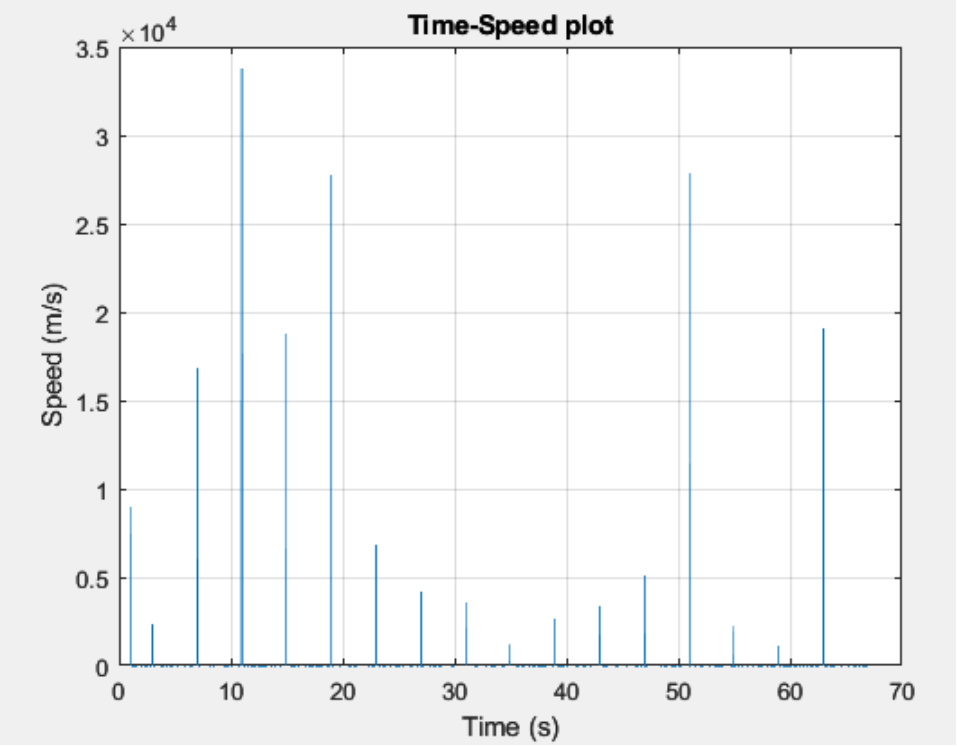


Figure 4.7: Time - Speed Graph

increase speed. This deviation offers a plausible explanation for the observed non-linearity in the trajectory towards the conclusion of the period under review.

4.2 Simulation-Based Performance Analysis

Due to a malfunction in the ROV's tether, the possibility of verifying the system's functionality in an actual aquatic environment was precluded. As a result, the assessment of the system's operational capabilities was limited to simulations conducted through ArduSub. These simulations, while not able to replicate the unpredictabilities of a real-world aquatic environment, provided insights into the system's theoretical performance.
However, it was possible to conduct tests in water to confirm the successful connection of the ROV with QGroundControl and ROS. This aspect of the system performed as expected, demonstrating reliable communication links. Furthermore, the localization process and data transmission were effectively tested in the water, affirming the system's ability to accurately determine its position and send this information.

To start the simulation, it is necessary to launch ROS TCP Endpoint, enabling communication between ROS and other software components, with the command:

Listing 4.1: Launch ROS TCP Endpoint

```
roslaunch ros_tcp_endpoint endpoint.launch
```

Then, a second terminal must be opened into the ardusub repository directory to launch the ArduSub Simulation, with the command:

Listing 4.2: Launch ArduSub simulation

```
sim_vehicle.py –L Marseille –S 1 —out=udp:0.0.0.0:14550
—map —console
```

This command starts the ArduSub simulation with the vehicle located at the port of Marseille. The -S 1 parameter sets the speed of the simulation. The –out parameter specifies the UDP output to which the simulation data is sent (port 14550), and –map –console opens additional interfaces for the map and console.
Finally, it is required to launch QGroundControl, the Unity Project and run bluerov_node.py code.

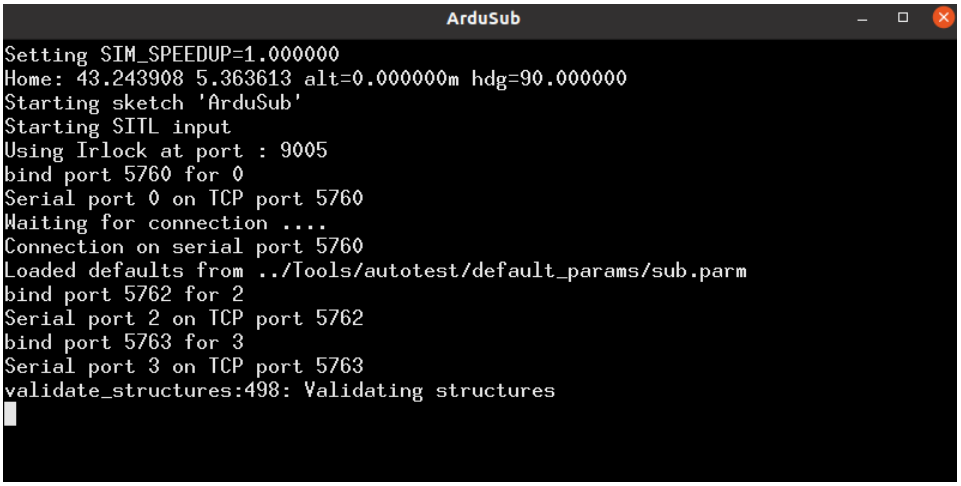
Figure 4.8 indicates the correct initialization of the ArduSub simulation. The system settings display the SIM_SPEEDUP parameter and Home location, confirming the simulation's starting position. The simulation is correctly connected with the MAVProxy module and it is notable the presence of a heartbeat from the simulated vehicle. In Figure 4.9, the console details the MAVProxy command line interface, which allows manual control and monitoring of the simulated vehicle's parameters such as battery level, heading, and altitude. The successful arming of the motors and setting of the vehicle mode indicates readiness for mission execution.

Figure 4.10 shows the ROS launch server output, confirming the successful launch of the Unity endpoint, which bridges the ROS and Unity environments. Meanwhile, in Figure 4.11, the Unity simulation environment set correctly up the local ROS IP.

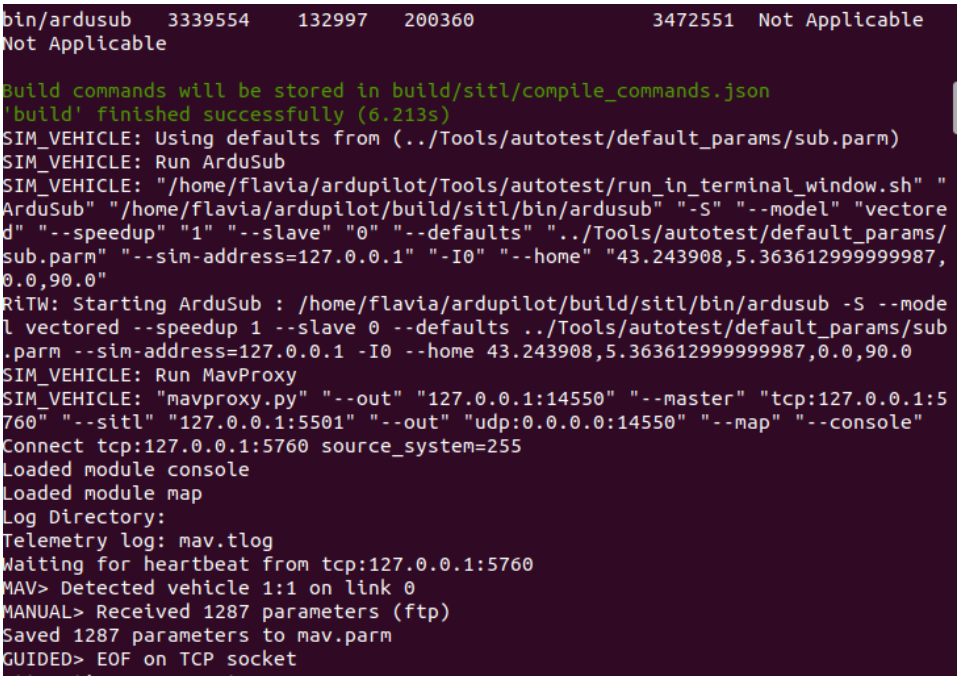
Figure 4.12 demonstrates the integration of the simulated vehicle within the QGC map interface with the correct localization in the port of Marseille.

The trajectory of the ROV is represented in Figure 4.13 (blue line). It shows a clear path with waypoints (green points) and points of interest, with target objects (red points).

Finally, Figure 4.14 provides a view of the mission logs, showing settings for a scanning mission, including depth, range sensor parameters and initial ROV position. The logs also indicate successful target detection and scanning operations, initially by reaching the desired points and identifying objects, and subsequently by calculating a new trajectory to revisit these points of interest.

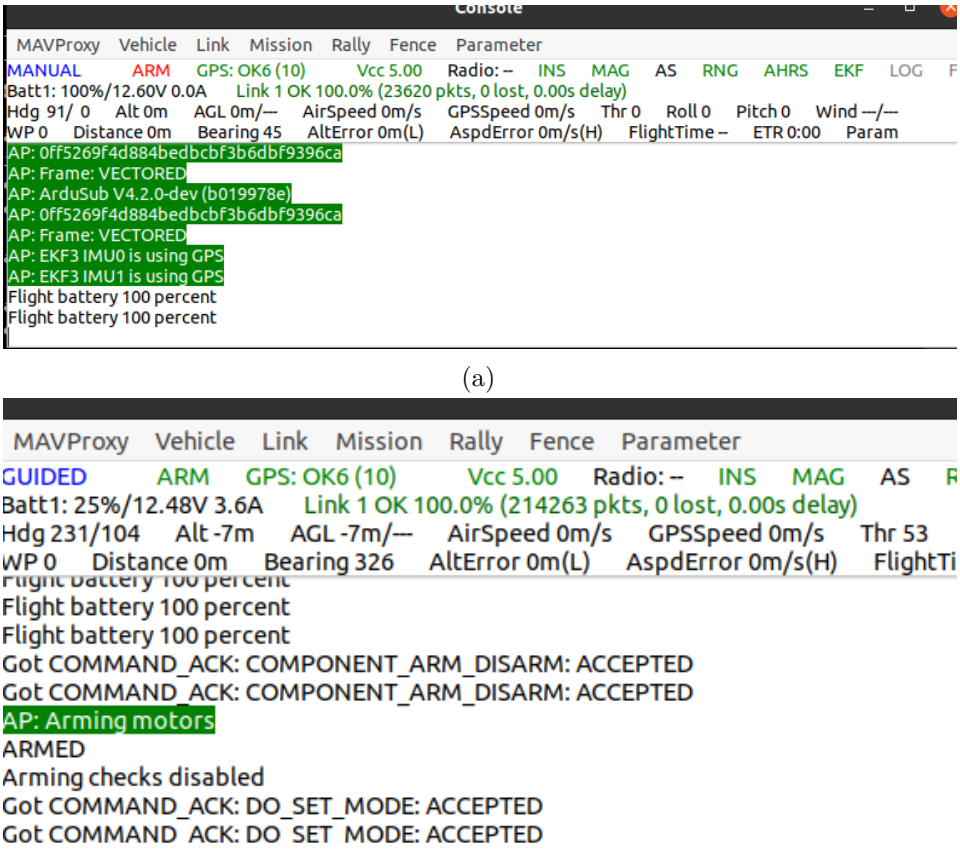


(a)



(b)

Figure 4.8: ArduSub Simulator: (a) parameters setting (b) terminal



(a)

(b)

Figure 4.9: MAVProxy Console

```
started roslaunch server http://flavia-VirtualBox:34099/

SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.16.0
 * /unity_endpoint/tcp_ip: 0.0.0.0
 * /unity_endpoint/tcp_port: 10000

NODES
  /
    unity_endpoint (ros_tcp_endpoint/default_server_endpoint.py)

auto-starting new master
process[master]: started with pid [2366]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to fc2333ce-9062-11ee-ad1f-7bb17f44181d
process[rosout-1]: started with pid [2376]
started core service [/rosout]
process[unity_endpoint-2]: started with pid [2379]
[INFO] [1701446532.122409]: Starting server on 0.0.0.0:10000
[INFO] [1701447032.899158]: Connection from 127.0.0.1
[INFO] [1701447033.009609]: RegisterSubscriber(/BlueRov2/local_position, <class
 'geometry_msgs.msg._PoseStamped.PoseStamped'>) OK
[INFO] [1701447033.048927]: RegisterSubscriber(/tf, <class 'tf2_msgs.msg._TFMes
sage.TFMessage'>) OK
```

Figure 4.10: ROS terminal

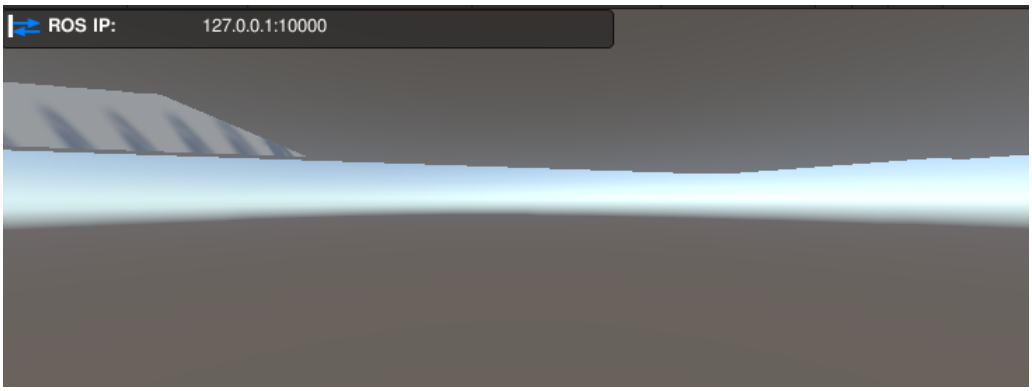


Figure 4.11: Unity-ROS connection

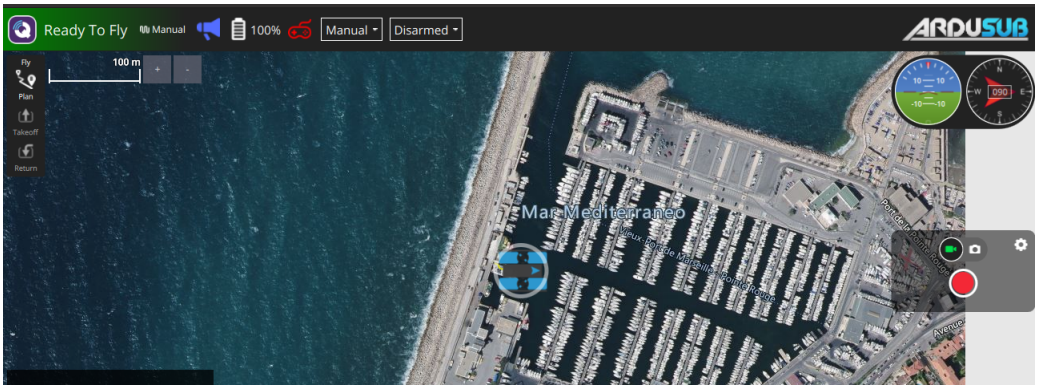


Figure 4.12: QGC localization map

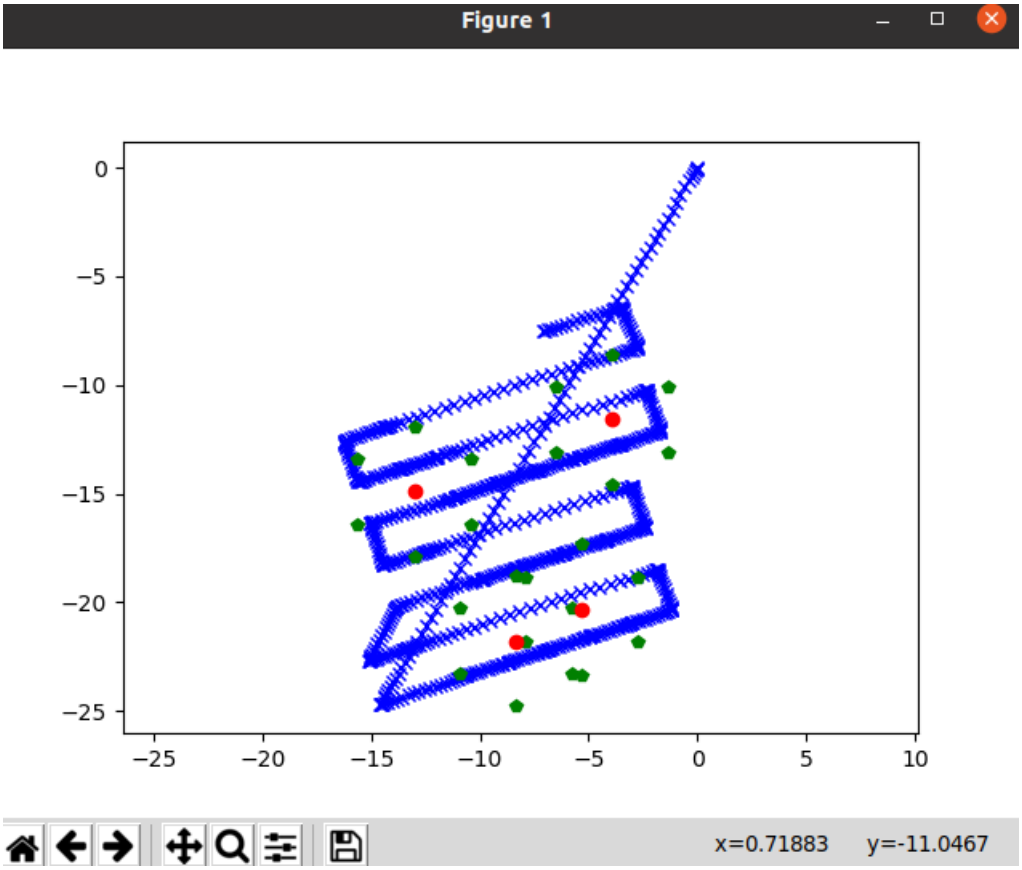
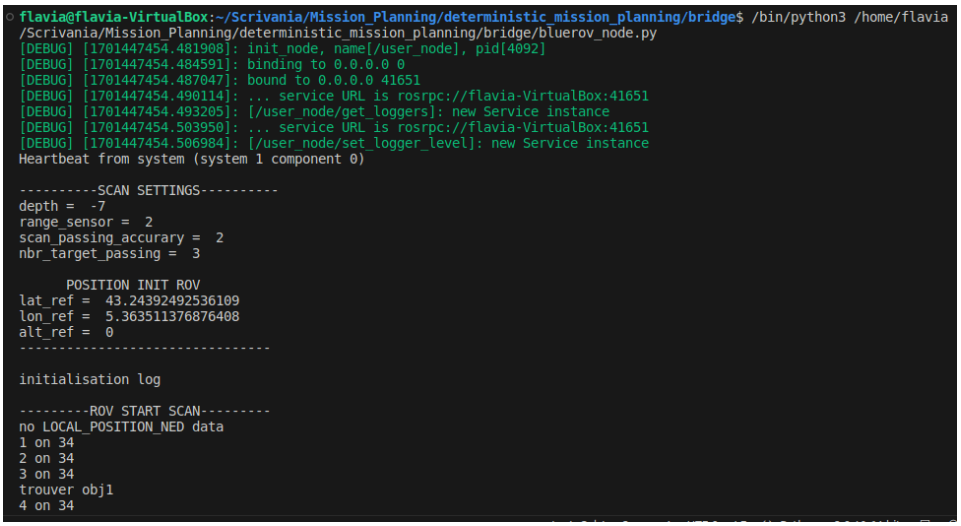
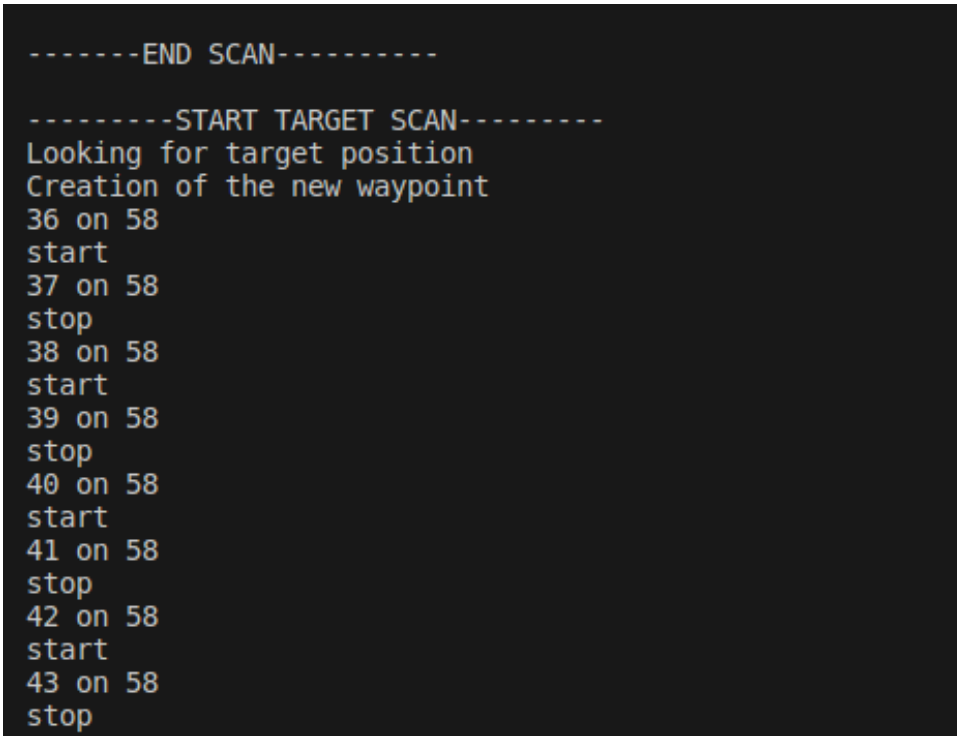


Figure 4.13: Rov's path



(a)



(b)

Figure 4.14: Python script terminal: (a) initialization, (b) new waypoints scanning

Conclusions

This thesis project, centered around the challenges of marine robotics, has culminated in an exploration of the field, integrating advanced technological solutions with practical applications. The selection of the BlueROV2 as the vehicle of choice, paired with the Seatrac USBL navigation system, was a strategic decision based on the project's requirements and the environmental challenges of underwater navigation. Implementing the Line of Sight control strategy with a linear trajectory proved to be highly effective for the project's objectives, ensuring precision and efficiency in the vehicle's movements.
Furthermore, the incorporation of a digital twin, mirroring the ROV's behavior in real-time using intermediary software such as ROS and Unity, represents an advancement in remote vehicle monitoring and control. This digital replication not only the system's functionality but also provided a tool for testing and simulation. The successful sea trial of the NGC system and the digital simulation using ArduSub firmware are proofs to the system's reliability and effectiveness.
The NGC system's performance during these tests demonstrated a high degree of control over the ROV's movements. It efficiently maintained stability, navigated to precise coordinates, and adapted its orientation in accordance with the mission's requirements. The data plots derived from these tests offer an analysis of the system's operational capabilities. They highlight not only the strengths of the system but also areas for potential improvement. For instance, further fine-tuning could minimize oscillations in the vehicle's pitch and optimize path efficiency, as seen in the variations in latitude and longitude. Such enhancements could lead to even more precise and efficient operations in future applications of marine robotics.

Moreover, this NGC system has some points of weakness, mostly dependent on the chosen hardware. Firstly, there is an inconsistent behavior of the *set_mode("ALT_HOLD")* function suggesting a possible firmware compatibility issue with the QGC version in use.
Then, the Seatrac system's limitation in providing accurate depth values under non-optimal environmental conditions was observed, indicating a sensitivity to external factors that could affect its reliability, like distances from surfaces or unknown interference.
Additionally, the Seatrac can transmit data at a minimum interval of every 4 seconds. While this delay did not pose a significant problem due to the low operational speed of the ROV, it is a factor that could limit the system's responsiveness in more dynamic

or demanding scenarios.
Furthermore, the inherent inaccuracies in the GPS and Seatrac systems, with potential errors up to two meters and one meter respectively, must be taken into account for precision operations.

In conclusion, future developments should focus on improving the positioning system by solving the identified issues.
Additionally, it is crucial to test the ROV system in conjunction with the digital twin in an actual marine environment and to develop a bidirectional communication between the two. This would enable not just data transmission from the ROV to Unity but also the sending of commands from the Unity simulated environment directly to the ROV.
A further advancement could involve adding a device for scanning points of interest in order to find objects, and incorporating this functionality into the existing algorithm.

Bibliography

[1] Rusty. What is an underwater rov? *BlueRobotics*, https://bluerobotics.com/learn/what-is-an-rov/.

[2] Liam Paull, Sajad Saeedi, Mae Seto, and Howard Li. Auv navigation and localization: A review. *IEEE JOURNAL OF OCEANIC ENGINEERING*, 39, JANUARY 2014.

[3] M. Fava. How much of the ocean has been explored? *Ocean Literacy Portal*, https://oceanliteracy.unesco.org/ocean-exploration/, 09 May 2022.

[4] R. Mendoza, D. Menacho, F. Cuellar, C. Carranza, and D. Arce. Multi-camera acquisition system for virtual model generation with underwater photogrammetry. *Conference Paper OCEANS*, San Diego, 2021.

[5] Dr. Mahizharuvi Poongundran, Prasannan. D, J. Sandra Agnes, Dr Saptorshi Das, D J ANUSHA, and Dina Amandykova. Role of underwater robots in ocean exploration research. *International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, Salem, India, 2022.

[6] Martina Rossi. Studio e sviluppo di algoritmi di controllo e documentazione ottica per veicoli sottomarin. *Università Politecnica delle Marche*, 2022.

[7] Xiang-Rui Huang and Liang-Bi Chen. An underwater explorer remotely operated vehicle. *IEEE Potentials*, 42, June 2023.

[8] Riccardo RUGGIU. Unmanned underwater navigation based on visual and inertial sensors. *Master's Degree Thesis, POLITECNICO DI TORINO*, April 2021.

[9] Rov types. *Hellenic Centre for Marine Research - ROV team*, 2020.

[10] Nicolò Ciuccoli. Intelligent systems for the exploration of structured and complex environments. *Ph.D. Dissertation, Università Politecnica delle Marche*, March 2019.

[11] Jianhua Bao, Daoliang Li, Xi Qiao, and Thomas Rauschenbach. Integrated navigation for autonomous underwater vehicles in aquaculture: A review. *Information Processing in Agriculture*, 2020.

[12] Xianbo Xiang, Lionel Lapierre, Bruno Jouvencel, and Guohua Xu. Cooperative acoustic navigation scheme for heterogenous autonomous underwater vehicles. *ResearchGate*, January 2009.

[13] Renzo Mendoza, Daniel Menacho, Francisco Cuellar, Cesar Carranza, and Diego Arce. Multi-camera acquisition system for virtual model generation with underwater photogrammetry. *OCEANS 2021*, an Diego – Porto, San Diego, CA, USA, 2021.

[14] Exploring our fluid earth, light in the ocean, https://manoa.hawaii.edu/exploringourfluidearth/physical/ocean-depths/light-ocean. *University of Hawaii*.

[15] Jules S. Jaffe. Underwater optical imaging: The past, the present, and the prospects. July 2015.

[16] Matthew Jhonson-Roberson et al. High-resolution underwater robotic vision based mapping and three-dimensional reconstruction for archaeology. *Wiley Periodicals*, 2016.

[17] Miguel Castillón et al. State of the art of underwater active optical 3d scanners. October 2019.

[18] Xiangxiong Kong and Ronny Garrett Hucks. Preserving our heritage: A photogrammetry-based digital twin framework for monitoring deteriorations of historic structures. *Automation in Construction*, 2023.

[19] A.E. Wright, D.L. Conlin, and S.M Shope. Assessing the accuracy of underwater photogrammetry for archaeology: A comparison of structure from motion photogrammetry and real time kinematic survey at the east key construction wreck. *Journal of Marine Science and Engineering*, 2020.

[20] Amine Mahiddine, Julien Seinturier, Daniela Peloso, Jean-Marc Boï, Pierre Drap, and Djamel Merad. Underwater image preprocessing for automated photogrammetry in high turbidity water. 2012.

[21] Likun Wang, Zi Wang, Peter Kendall, Kevin Gumma, Alison Turner, and Svetan Ratchev. Digital-twin deep dynamic camera position optimisation for the v-starsphotogrammetry system based on 3d reconstruction. *NTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 2023.

[22] Massot-Campos Miquel and Gabriel Oliver-Codina. Optical sensors and methods for underwater 3d reconstruction. *Sensors*, 2015.

[23] Nuno Gracias, Rafael Garcia, Ricard Campos, Natalia Hurtos, Ricard Prados, ASM Shihavuddin, Tudor Nicosevici, Armagan Elibol, Laszlo Neumann, and Javier Escartin. Application challenges of underwater vision. February 2017.

[24] Abdou Shalaby, Mohammed Elmogy, and Ahmed Abo El-Fetouh. Algorithms and applications of structure from motion (sfm): A survey. *International Journal of Computer and Information Technology*, November 2017.

[25] MathWorks. Che cos'è il digital twin? *https://it.mathworks.com/discovery/digital-twin.html*.

[26] Adam Thelen, Xiaoge Zhang, Olga Fink, Yan Lu, Sayan Ghosh, Byeng D. Youn, Michael D. Todd, Sankaran Mahadevan, Chao Hu, and Zhen Hu. A comprehensive review of digital twin - part 1: Modeling and twinning enabling technologies. 2022.

[27] Mohsen Attaran and Bilge Gokhan Celik. Digital twin: Benefits, use cases, challenges, and opportunities. *Decision Analytics Journal*, 2023.

[28] Thor I. Fossen. *HANDBOOK OF MARINE CRAFT HYDRODYNAMICS AND MOTION CONTROL*. John Wiley & Sons Ltd, 2011.

[29] Garcia-Valdovinos Luis et al. Model-free high order sliding mode control with finite-time tracking for unmanned underwater vehicles. *Applied Sciences*, 2021.

[30] BlueRobotics. Navigator flight controller. *https://bluerobotics.com/store/comm-control-power/control/navigator/*.

[31] BlueRobotics. Bluerov2 datasheet. *https://bluerobotics.com/store/rov/bluerov2/*.

[32] BlueRobotics. Fathom rov tether (rov-ready). *https://bluerobotics.com/store/cables-connectors/cables/fathom-rov-tether-rov-ready/*.

[33] Seascape Subsea BV. X100 series micro-usbl tracking, data modem & ahrs acoustic beacons, user manual.

[34] Seascape Subsea BV. Seatrac x150 – usbl beacon. *https://www.seascapesubsea.com/product/seatrac-x150/*.

[35] Seascape Subsea BV. Seatrac x010 – modem beacon. *https://www.seascapesubsea.com/product/seatrac-x010-modem-beacon/*.

[36] BlueRobotics. Outland technology power supply for the bluerov2. *https://bluerobotics.com/store/comm-control-power/powersupplies-batteries/otps1kw/*.

[37] ArduSub. Overview. *https://www.ardusub.com/*.

[38] MAVLink. Mavlink developer guide. *https://mavlink.io/en/*.

[39] Dronecode. Qgroundcontrol user guide. *https://docs.qgroundcontrol.com/master/en/index.html*.

[40] ShaneLoretz. Ros noetic ninjemys. *http://wiki.ros.org/noetic*.

[41] Unity engine. *https://unity.com/products/unity-engine*.

[42] Gangwen Wei and Jie Yang. Path following optimization of unmanned ships based on adaptive line-of-sight guidance and deep q-network. *International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, 2022.

[43] Khoshnam Shojaei and Mehdi Dolatshahi. Line-of-sight target tracking control of underactuated autonomous underwater vehicles. *Ocean Engineering*, 2017.

[44] Andreas Andreou, Constandinos X. Mavromoustakis, Jordi Mongay Batalla, Evangelos K. Markakis, George Mastorakis, and Shahid Mumtaz. Uav trajectory optimisation in smart cities using modified a* algorithm combined with haversine and vincenty formulas. *IEEE Transactions on Vehicular Technology*, 2023.