POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master Thesis

Motion Control of an Autonomous Underwater Vehicle for Maritime Surveillance

Supervisors

Prof. Diego REGRUTO TOMALINO

Cosupervisor

Prof. Roberto GALEAZZI

Candidate

Anastasia Costanza AIASSA

APRIL 2025

To my mom and dad,
who made all of this possible.

Motion Control of an Autonomous Underwater Vehicle for Maritime Surveillance

Anastasia Costanza Aiassa

Abstract

Recent geopolitical events in Europe have increased the need to focus on securing critical infrastructure, such as energy, communications, and transportation networks. A significant portion of this critical infrastructure is situated in marine environments, such as ports that facilitate goods transport, offshore energy installations, and underwater cabling.

Autonomous Underwater Vehicles (AUVs) present a threat to maritime critical infrastructure, since they can approach these resources undetected, potentially causing significant damage. Motivated by this, there is a growing interest in developing Autonomous Underwater Interception Drones (AUID), capable of intercepting and tracking an intruding AUV in complex marine environments.

This thesis explores the fundamental steps for the design and development of the motion control system for an AUID, addressing both motion planning and control. The operational environment is defined as an underwater windfarm, where obstacles are static and predetermined. Therefore, during motion planning, an obstacle avoidance algorithm is employed to navigate the known obstacles. For the motion control phase, a control technique based on Control Barrier Functions (CBFs) is integrated with a nominal controller to mitigate the effects of potential disturbances, such as e.g. ocean currents, ensuring precise trajectory tracking in uncertain scenarios.

Numerical results providing an appraisal of the overall controller performance and behaviour are presented using a benchmark system, showing significant promise for a subsequent experimental phase. The proposal within this project can be used as a fundamental stepping stone towards an effective intruder detection and interception, providing the basis for planning and robust tracking in this family of devices.

Keywords: Autonomous Underwater Interception Drone (AUID), Control Barrier Functions (CBF), RRT*, LOS Guidance.

Acknowledgments

First and foremost, I would like to extend my deepest gratitude to Nicolas Faedo for giving me the opportunity of this challenge. Thank you for your constant availability, kindness, presence, dedication and professionalism you have shown me over these months. You have been my greatest source of guidance, and I can not thank you enough for that.

I would like to thank Professor Regruto for giving me this opportunity and for the remote guidance and assistance. Furthermore, my gratitude goes to Professor Galeazzi for allowing me to carry out this work in such a dynamic and inspiring environment as the Technical University of Denmark (DTU).

I would like to sincerely thank all the people I have had the privilege of meeting during my years at the Politecnico. To those who shared lectures, laughs, and stress, thank you for walking this path with me. It would not have been the same without you.

To the PoliTOcean Team, thank you not just for introducing me to the marine robotics world, but for making it an unforgettable journey. This experience has become a part of who I am, and I will carry its lessons and laughter with me well beyond these pages.

Finally, to my family, my source of strength and inspiration. Thank you for your constant support and guidance throughout these years. None of this would have been possible without your encouragement and presence. Every achievement I reach is dedicated to you.

Table of Contents

List of Figures

List of Tables

Acronyms

GNC	Guidance, Navigation and Control.
DOF	Degree of Freedom.
DOFs	Degrees of Freedom.
AUV	Autonomous Underwater Vehicle.
AUID	Autonomous Underwater Interception Drone.
ROV	Remotely Operated Vehicle.
RRT	Rapidly Exploring Random Tree.
RRT*	Rapidly Exploring Random Tree Star.
LOS	Line-of-Sight.
TAM	Thruster Allocation Matrix.
CLF	Control Lyapunov Function.
CBF	Control Barrier Function.
PID	Proportional-Integral-Derivative.
LQR	Linear Quadratic Regulator.
ACC	Adaptive Cruise Control.
LK	Lane Keeping.
CBF-QP	Control Barrier Function - Quadratic Problem.
CLF-CBF-QP	Control Lyapunov Function - Control Barrier Function - Quadratic Problem.
ROS	Robot Operating System.

Chapter 1

Introduction

This chapter introduces the M.Sc. thesis project entitled *Motion Control for an Autonomous Underwater Vehicle for Maritime Surveillance*. In particular, Section 1.1 presents the background and motivation which gives rationale to this thesis. Section 1.2 presents the problem statement and overall project objectives, followed by an overview of the methods and tools employed in Section 1.3. Finally, Sections 1.4 and 1.5 summarize the main contributions and outline the remaining chapters.

1.1 Background and Motivation

Recent geopolitical events in Europe have highlighted the need to improve the security of critical infrastructure. Several nations are focusing on protecting key infrastructure like energy networks, communication systems, and transport hubs. Many of these essential infrastructures are underwater, including ports, which play a crucial role in global trade, offshore energy platforms, and the vast network of undersea cables and pipelines.

Meanwhile, the use of Autonomous Underwater Vehicles (AUVs) has grown significantly. This is a key resource for environmental monitoring, underwater exploration, and infrastructure inspection. Their ability to work independently in complex marine environments makes them ideal for tasks that can be risky for manned missions. However, this also poses a significant threat, as AUVs can enter protected zones unnoticed, potentially causing serious damage. Addressing such intrusions is extremely challenging, especially since operations must also ensure that assets remain secure.

Recent events like the sabotage of the *Nord Stream* pipelines in 2022 have highlighted how vulnerable underwater facilities can be. This has led governments to increase security efforts. According to [2], protecting critical maritime infrastructure is especially challenging due to the vastness of oceans, the complex underwater environment and the presence of different industries including transport, energy, communications, fishing, and biodiversity. The same research also emphasizes that the definition of "critical infrastructure" is more of a political manner, requiring

Figure 1.1: An Autonomous Underwater Vehicle being deployed [1].

coordination between multiple areas, such as defense, maritime safety, and cyber security, to build resilience. In this context, the Nord Stream attack was described as a “wake-up call” [3], exposing weaknesses in current defenses. As a result a stronger awareness for the need of better security in maritime systems is rapidly growing.

To address this growing challenge, it is becoming increasingly important to develop countermeasures that can effectively deal with unauthorized underwater activities. One promising solution is the design and deployment of Autonomous Underwater Interception Drones (AUIDs). These drones should be specifically studied to track and intercept intruding AUVs in potentially complex marine environments. Among the main objectives these devices shall perform are real-time surveillance, the protection of essential underwater infrastructure, and rapid adaptation to new threats.

1.2 Problem Statement and Project Objectives

Motivated by the underlying importance of AUIDs, this project aims to design and develop a motion control system, which addresses both the planning and the motion control of the vehicle itself, including collision avoidance. The study begins by developing a trajectory planning algorithm that, by using the available information within the operation environment, such as offshore platforms, allows the AUID to quickly intercept an intruding moving AUV, once detected by an active acoustic system. To handle disturbances such as sudden currents, a nominal control system is implemented with an additional safety layer using Control Barrier Functions (CBFs). This combined approach ensures that the AUID accurately follows its desired trajectory despite the influence of such disturbances.

Note: The terms Unmanned Underwater Vehicle (UUV), Autonomous Underwater Vehicle (AUV), and Remotely Operated Underwater Vehicle (ROV) are interchangeably used in this project.

1.3 Methods and Tools

The implementation has been conducted primarily on a 64-bit Ubuntu 20.04.6 LTS operating system, using the Robot Operating System (`ROS`) and `Gazebo 11` for simulation.

To model and control the AUV, the open source simulation framework *uuv_simulator* was employed. This simulator extends ROS and Gazebo with plugins and models specifically useful for unmanned underwater vehicles. These include modules that replicate underwater physics, such as buoyancy, drag, and currents. The framework is actively maintained on GitHub (see [4] for more details). The system used throughout the project is the BlueROV2, developed by *BlueRobotics*.

Subsequently, the second phase of the study, focusing on CFBs, is carried out using `MATLAB R2023b` to ensure a more detailed analysis of their behavior within the overall control framework.

1.4 Contributions

The main contributions of this thesis can be summarized as follows:

- Providing a short overview of the *Motion Planning* and *Motion Control* phases for *Autonomous Underwater Vehicles*. An implementation in simulation of each phase of the motion control system for an autonomous vehicle has been developed, including path planning, a guidance system, and a control system, aiming to evaluate how these modules interact and fulfil their functions. It is important to note that the navigation module has been excluded from this phase of the study, since the focus is primarily on the control and guidance aspects necessary for autonomous motion execution.

- Implementation of a 2D *Autopilot* on a BlueROV2 underwater vehicle, using the `ROS/Gazebo` simulation environment. This implementation allows analyzing the outcome of the simulation in a realistic scenario.

- Study and implementation of a control strategy based on *Control Barrier Functions*, applied in terms of a *Lane-Keeping* problem to ensure path following within predefined boundaries.

1.5 Thesis Outline

The content of the project is organized into 8 Chapters and structured as follows:

- **Chapter 2:** A brief analysis of the state-of-the-art in *Guidance, Navigation, and Control (GNC)* in a marine environment is given, followed by a review of both path planning and guidance strategies in the marine field. Finally, a review of the implementation of CBFs in autonomous systems is discussed.

- **Chapter 3:** The mathematical modelling for the AUV under study is presented, outlining the formulation of the dynamic model and the derivation of the control model necessary for the autopilot.

- **Chapter 4:** The algorithm used to generate the desired path for the AUV is detailed, providing a comprehensive description of the implementation of a Rapidly Exploring Random Tree Star (RRT*) algorithm, including a discussion on the chosen parameters.

- **Chapter 5:** The guidance strategy used to direct the AUV along the planned path is presented, providing a comprehensive description of the implemented Line-of-Sight (LOS) guidance method, including a discussion on the chosen parameters.

- **Chapter 6:** The design and implementation of the autopilot system are explained, covering both the speed and steering control mechanisms. A CFB-based approach to maintain a correct path following, even with the presence of disturbances, is given.

- **Chapter 7:** The simulation setup used to test the proposed system is described. The simulation environment and analysis of the results are discussed.

- **Chapter 8:** A conclusion summary of the study and potential future directions are discussed.

Chapter 2

Literature Review

This chapter presents an analysis of the state-of-the-art in *GNC* systems applied to *AUVs*. In the first part (Section 2.1), an overview of path planning and guidance in the marine field is presented. The second part of this chapter, described within Section 2.4, discusses the main elements comprising the state-of-the-art regarding *CBF* applied in autonomous systems.

2.1 Guidance, Navigation and Control

According to [5], a motion control system for AUVs is generally structured into three blocks: Guidance, Navigation, and Control. The three modules interact with each other, as shown in Figure 2.1.

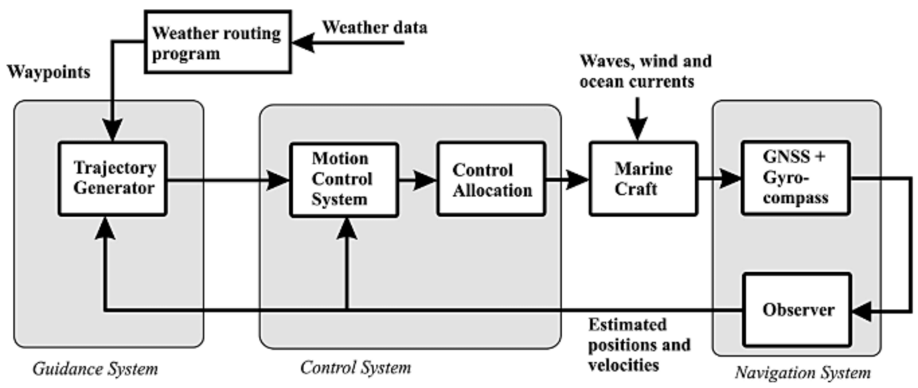

Figure 2.1: GNC module interaction([5], p. 233)

In particular, each module operates as follows:

Guidance: This system has the objective of computing the desired reference position, velocity, and acceleration of the vehicle.

Navigation: This task involves determining the vehicle's position, orientation, course, and distance travelled. It can also involve the calculation of velocity and acceleration. This is achieved using systems like GNSS (Global Navigation Satellite System) in combination with motion sensors such as accelerometers and gyroscopes.

Control: This block involves the calculation of the forces and moments required to achieve specific control tasks, such as setpoint regulation, trajectory tracking, or path-following.

2.1.1 Path Planning and Guidance

As discussed in [6], path planning and guidance are two interconnected components of a motion control system. While path planning is responsible for generating a safe and feasible path for the vehicle to follow, the guidance computes the reference trajectories required to guide the vehicle along the planned path. In other words, path planning determines *what* the vehicle should achieve, meaning a particular position, while guidance determines *how* the vehicle should move to reach the point, by generating appropriate reference trajectories. Figure 2.2 illustrates the interaction among all the complete modules.

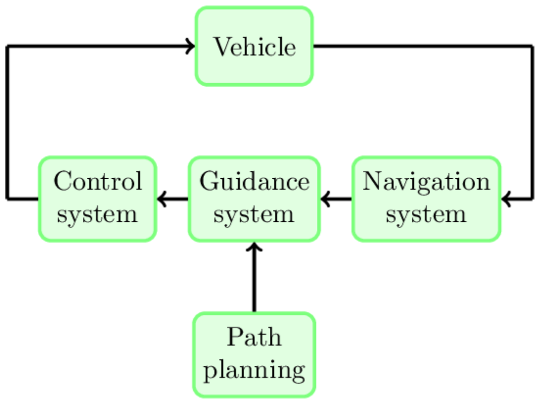

Figure 2.2: Interaction between GNC and Path Planning ([6], p. 5)

2.2 Path Planning

Path planning is the process of determining a safe route from a starting point to a goal while avoiding collisions. For AUVs, this is often difficult due to complex underwater environments, with moving obstacles and movement limitations. Path planning methods are generally divided into two types [7]: *global path planning* and *local path planning*. The first one uses prior knowledge of fixed obstacles to create a feasible path, while the second one adjusts the path in real-time, based on sensor data, while avoiding unexpected obstacles.

Path planning techniques, specifically, are divided based on their approach to finding the path, being grouped into deterministic and probabilistic algorithms. Deterministic methods always produce the same path under the same conditions. They either find a solution, if one exists, or they ensure that no solution is available. Common deterministic techniques are Artificial Potential Fields (APF), Cell Decomposition

[8], Voronoi diagrams [9] and the so-called A* Algorithm [10].

Probabilistic methods find a solution, if one exists, but can produce different paths each time, even under similar conditions. These methods can be divided also into two categories: sampling techniques and diffusion techniques. Sampling techniques work by examining the environment to create a map that can be used again, this is called a roadmap. A common technique is the Probabilistic Roadmap (PRM) [11]. Diffusion techniques, on the other hand, explore the space randomly until they find, if exists, the solution. One of the most popular algorithms of this type is the Rapidly-exploring Random Trees (RRT) algorithm. It works by randomly sampling the space and constructing a tree of feasible paths. This algorithm works well even in complex environments and without performing any preprocess of the environment, which makes it good for real-time applications. However, since RRT produces non-optimal solutions, [12] proposed a modification of RRT called RRT*. This new algorithm can find optimal paths, a feature which was not guaranteed in standard RRT.

When dealing with dynamic targets, RRT* presents, in general, good behaviour. For instance, in [13], an RRT* based approach is proposed in an AUV that needs to compute a rendezvous with a moving surface target. The AUV dynamically updates the trajectory when pursuing the moving recovery vessel. This approach recalculates the planned trajectory in real-time, based on the information on the AUV's current position. Similarly, [14] introduced an RRT* based approach in an AUV to track and reach a dynamic target using only angle-based measurements. Their results showed that RRT* can achieve good performance in target tracking, with moderate computational complexity.

2.3 Guidance

Guidance, as defined by [15], is "*the process of steering an object along a path toward a target point, which may be moving*". In control theory, motion control scenarios are generally categorized into *setpoint regulation*, *trajectory tracking*, and *path following*. The first one consists in stabilizing a system at a constant position and orientation, the second one guarantees the system to follow a reference trajectory in a given time, while the third one, path following, focuses on guiding the system along a path without time constraints, which allow the vehicle to move with different speed while avoiding obstacles. To do so, several guidance laws have been adopted from the missile community, which is one of the most active in the field of guidance research. The main guidance laws that have been adopted in the marine field are: Line-of-Sight (LOS), Pure Pursuit (PP), and Constant Bearing (CB) guidance. These laws have the goal of directing a vehicle's motion toward a target. The differences between them are analyzed in detail in [16], Chapter 10.

One of the most adopted methods for path following in underwater vehicles is the LOS guidance [17] [18]. A critical parameter in LOS is the lookahead distance, which is the distance ahead of the vehicle along the desired path at which a target point is selected. These values can deeply change the motion of the vehicle since smaller values can cause aggressive steering yet cause oscillations, while larger values allow a smoother motion but with a slower convergence. Different studies on the lookahead distance values have been done, such as time-varying lookahead distances to balance these trade-offs, as shown in [19]. A drawback of this guidance is its difficulty in handling disturbances, such as the current. To address this issue, some studies have been conducted, as reported in [16], where an integral action is introduced into the guidance law to mitigate the steady-state error caused by such forces.

2.4 Control Barrier Function

The application of safety-critical control is becoming increasingly important in autonomous systems. In such systems, it is not enough to simply ensure a stability condition, which means, to reach a desired state for the system, but a safety condition must also be guaranteed. In other words, while stability implies that the system reaches a desired condition over time, safety ensures that it avoids dangerous situations. These concepts of stability and safety are formally represented by CLF and Control Barrier Function CBF, as illustrated by [20].

As discussed in [20], the term ”barrier” comes from the optimization field, where barrier functions are added to cost functions to steer the solution away from undesirable areas. When a system has a control input, this concept is extended to create a ”Control Barrier Function”. A CBF is designed to be positive inside the safe region, zero on the boundary, and negative outside. Also, for every state within the safe region, there is always a control input that will keep the system safe in that region.

One way to ensure safety is the so-called “Lyapunov-like” approach. In simple terms, these functions create invariant level sets, and if these sets remain inside a designated safe region, safety is ensured. One particular technique called CBF-QP, uses quadratic programming to make the smallest possible change to a (nominal) feedback controller in order to maintain and guarantee safety [21].

CBFs are widely used for obstacle avoidance in dynamic environments, in particular during trajectory tracking. Some works are presented in [22], [23] and [24]. Another application is the ACC, where they are used to ensure safety while keeping the vehicle maintaining a fixed cruising speed. Whenever a moving vehicle is detected, the controller reduces the speed of the vehicle to keep a predefined distance of safety. This problem was presented in [25] and experimental tests were conducted in [26], where this approach successfully handled safety, speed regulation and comfort.

Similarly, LK systems use CBF to ensure the vehicle stays ”centered” within its lane even when incurring in a possible curved lane [27], [28]. In the context of autonomous underwater vehicles, this application is critical because sudden currents or disturbances can force the vehicle to deviate from its defined trajectory, exceeding some predefined bounds of safety.

Recent studies have explored the use of barrier functions not only in the path-following phase but also in the motion-planning stage. The first CBF-based sampling algorithm [29] eliminated explicit collision checking by using CBFs' forward invariance. Later works, such as Adaptive CBF-RRT* [30], introduced a rewiring step to optimize trajectories but a drawback of this approach was the computational costs that were too high due to iterative QP solving. Other studies have been conducted incorporating an LQR control. In fact, the LQR-CBF-RRT* [31] framework integrates CBFs in the steering phase of RRT* and an LQR controller generates an optimal control sequence to steer the trajectory while ensuring, at each step, that CBF constraints enforce safe motion. This guarantees not only efficient trajectory generation but also real-time safety preservation during the path execution. Lastly, other studies have integrated RRT* with a control phase based on CBF and CLF during the expansion of new nodes by introducing a function, the *compatibility* function [30]. This function checks whether a CLF-CBF-based controller can successfully steer the system from one waypoint to another, guaranteeing a feasible and optimal trajectory.

The growing use of CBFs in both motion planning and path following highlights their crucial role in guaranteeing the safety of autonomous systems while ensuring stability conditions, a condition of great importance in these kinds of systems.

Chapter 3

Mathematical modelling

This section introduces the specific ROV considered within this project and the fundamentals for its mathematical modelling. In particular, following the notation introduced in Section 3.1, the overall dynamical modelling procedure is discussed within Sections 3.2, 3.3 and 3.4, while the specific parameters used for the ROV considered in this thesis are listed in Section 3.5.

The platform used in this project is the *BlueROV2*, developed by *Blue Robotics Inc.* The *BlueROV2* is an open-source underwater vehicle. The ROV can be seen below in Figure 3.1.

Figure 3.1: BlueROV2 by *Blue Robotics Inc* - Figure adapted from [32].

The mathematical notations and equations of motion used in this thesis are based on the modelling framework described by Fossen in [5]. Fossen's methodology for describing the dynamics of underwater vehicles provides a relatively straightforward way to represent all six degrees-of-freedom (DOFs).

Within this chapter, the simulation (referred to as the "*high-fidelity*") mathematical model of the BlueROV2 is derived, representing the most accurate approximation of the real system considered in this thesis. This model is used in the simulation phase to replicate the behavior of the real system to be controlled. Due to the complexity of the full (simulation) model, approximations are made to design a simplified model-based control system.

3.1 Notation

An AUV operates with six DOFs, which can be described using a vectorial representation. This follows the *SNAME (Society of Naval Architects and Marine Engineers - 1950)* notation, as detailed in Table 3.1, where six generalized coordinates are used to define the position and orientation of the vehicle.

DOF	**Description**	**Forces and moments**	**Linear and angular velocities**	**Positions and Euler angles**
1	Surge:motion in the x direction	X	u	x
2	Sway:motion in the y direction	Y	v	y
3	Heave:motion in the z direction	Z	w	z
4	Roll:rotation about the x axis	K	p	ϕ
5	Pitch:rotation about the y axis	M	q	θ
6	Yaw:rotation about the z axis	N	r	ψ

Table 3.1: 6-DOFs states defined according to SNAME 1950 [33].

Considering the BlueROV2, the vehicle is equipped with six thrusters: four of them enable movements in *surge*, *sway*, and *yaw* directions, while the other two are used for *heave* and *pitch*. Due to the specific arrangement of the thrusters, as illustrated in Figure 3.2, *roll* motion cannot be performed with this configuration.

3.1.1 Coordinates

According to the SNAME notation (Table 3.1), the generalised pose, velocity and forces and moments coordinates can be written, in compact form, as

$$\boldsymbol{\eta} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T, \tag{3.1}$$

$$\boldsymbol{\nu} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T, \tag{3.2}$$

$$\boldsymbol{\tau} = \begin{bmatrix} X & Y & Z & K & M & N \end{bmatrix}^T. \tag{3.3}$$

The following sub-vectors are also given for convenience of notation:

- **Position:**
$$\boldsymbol{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T \in \mathbb{R}^3,$$

- **Euler angles:**
$$\boldsymbol{\Theta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T \in SO(3),$$

- **Linear velocity:**
$$\boldsymbol{v} = \begin{bmatrix} u & v & w \end{bmatrix}^T \in \mathbb{R}^3,$$

- **Angular velocity:**
$$\boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^T \in \mathbb{R}^3,$$

- **Force:**
$$\boldsymbol{f} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T \in \mathbb{R}^3,$$

- **Moment:**
$$\boldsymbol{m} = \begin{bmatrix} K & M & N \end{bmatrix}^T \in \mathbb{R}^3,$$

where $\mathbb{R}^3$ denotes the three dimensional Euclidean space and $SO(3)$ indicates the three dimensional sphere in which three angles are defined on the interval of $[-\pi, \pi]$ for ϕ and ψ, and the interval of $[-\pi/2, \pi/2]$ for θ.

Therefore, the general motion of an AUV in 6 DOFs can be described by the following vectors:

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{\Theta} \end{bmatrix} \in \mathbb{R}^3 \times SO(3), \tag{3.4}$$

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} \in \mathbb{R}^6, \tag{3.5}$$

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{m} \end{bmatrix} \in \mathbb{R}^6, \tag{3.6}$$

where $\boldsymbol{\eta}$ is the position and orientation (alse called pose) vector, $\boldsymbol{\nu}$ is the linear and angular velocity vector and $\boldsymbol{\tau}$ is the force and moment vector.

3.2 BlueROV2 Dynamics

The dynamics of an underwater vehicle can be divided into two main components: *kinematics*, which deals with the geometry of motion, and *kinetics*, which explains how forces and moments affect the motion. Regarding the former, the general 6 DOF equation is developed while, for the latter, a 6 DOF model is implemented in Gazebo (see Section 7.1.1 for further details) but for control purposes a 3 DOF linearized model is derived.

3.2.1 Kinematics

When analyzing the motion of marine craft in 6 DOFs, it is convenient to define two important reference frames, as can be seen in Figure 3.2. These are defined explicitly in the following paragraphs.

NED Frame: The *North-East-Down* (NED) coordinate system $\{n\} = (x_n, y_n, z_n)$, is based on the Earth's reference ellipsoid (World Geodetic System, 1984) and has its origin at the craft's location. The NED frame is tangent to the Earth's surface

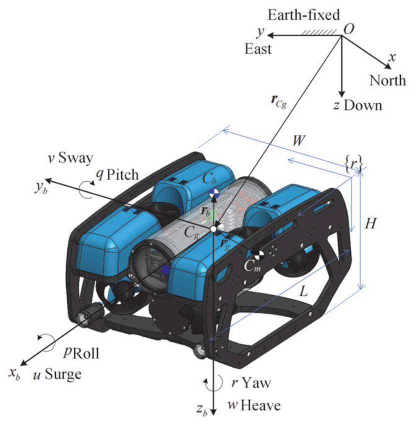

Figure 3.2: The BlueROV2 coordinates [34].

and moves with the craft. The x-axis points towards true North, the y-axis points East, and the z-axis points downward, normal to the Earth's surface.

BODY Frame: The body-fixed coordinate system, denoted as $\{b\} = (x_b, y_b, z_b)$, is fixed to the craft and moves with it. The origin, o_b, is located at the craft's center. The body frame defines the position and orientation of the AUV relative to the inertial NED frame. Linear and angular velocities are expressed in the body-fixed frame.

The rotation matrix from frame $\{b\}$ to frame $\{n\}$ can be described by a sequence of three principal rotations about the z, y, and x axes, also known as *zyx-convention*. In particular, the rotation matrix is defined as

$$\mathbf{R}_b^n(\mathbf{\Theta}_{nb}) := \mathbf{R}_{z,\psi}\,\mathbf{R}_{y,\theta}\,\mathbf{R}_{x,\phi} = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi c_\phi s_\theta \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}, \quad (3.7)$$

where $s\cdot := \sin(\cdot)$ and $c\cdot := \cos(\cdot)$. Here, the vector of Euler angles is $\mathbf{\Theta} nb = [\phi,\ \theta,\ \psi]^\top$, with ϕ representing roll, θ pitch, and ψ yaw.

The body-fixed *velocity vector* $\mathbf{v}_{b/n}^b := [\,u,\ v,\ w\,]^\top$ is related to the earth-fixed velocity vector $\dot{\boldsymbol{p}}_{b/n}^n := [\dot{x},\ \dot{y},\ \dot{z}]^\top$ through the rotation matrix in Equation (3.7) as follows:

$$\dot{\boldsymbol{p}}_{b/n}^n = \mathbf{R}_b^n(\mathbf{\Theta}_{nb})\,\mathbf{v}_{b/n}^b.$$

While, the body-fixed *angular velocity vector* $\boldsymbol{\omega}_{b/n}^b := [\,p,\ q,\ r\,]^\top$ is connected to the Euler angle rate vector $\dot{\mathbf{\Theta}}_{nb} := [\,\dot{\phi},\ \dot{\theta},\ \dot{\psi}\,]^\top$ via the transformation matrix $\mathbf{T}_\Theta(\mathbf{\Theta}_{nb})$

$$\boldsymbol{T}_\Theta(\mathbf{\Theta}_{nb}) = \frac{1}{c_\theta}\begin{bmatrix} c_\theta & s_\phi s_\theta & c_\phi s_\theta \\ 0 & c_\phi c_\theta & -s_\phi c_\theta \\ 0 & s_\phi & c_\phi \end{bmatrix} \quad (3.8)$$

as follows:

$$\dot{\mathbf{\Theta}}_{nb} = \mathbf{T}_{\Theta}(\mathbf{\Theta}_{nb})\,\boldsymbol{\omega}^b_{b/n}.$$

Notice that the transformation matrix $\boldsymbol{T}_{\Theta}$ in (3.8) becomes singular as θ approaches $\pm\pi/2$. However, in normal operations, the pitch angle θ is assumed to be small, so this singularity is not problematic.

By combining the transformations for *linear* and *angular velocities*, the general 6 DOF kinematic equations can be expressed in vector form as

$$\dot{\boldsymbol{\eta}} = \mathbf{J}_{\Theta}(\boldsymbol{\eta})\,\boldsymbol{\nu} \quad \iff \quad \begin{bmatrix} \dot{\boldsymbol{p}}^n_{b/n} \\ \dot{\mathbf{\Theta}}_{nb} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^n_b(\mathbf{\Theta}_{nb}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{T}_{\boldsymbol{\Theta}}(\mathbf{\Theta}_{nb}) \end{bmatrix} \begin{bmatrix} \mathbf{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix}. \tag{3.9}$$

When the roll ϕ and pitch θ angles are small, it is possible to approximate the rotation and transformation matrices as

$$\mathbf{R}^n_b(\mathbf{\Theta}_{nb}) \approx \mathbf{R}^n_b(\psi) = \mathbf{R}_{z,\psi} = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.10}$$

$$\mathbf{T}_{\Theta}(\mathbf{\Theta}_{nb}) \approx \mathbf{I}_{3\times3}, \tag{3.11}$$

where $R_{z,\psi}$ is the rotation matrix corresponding to a rotation about the z-axis, and $I_{3\times3}$ is the identity matrix in $\mathbb{C}^{3\times3}$.

3.2.2 Kinetics

Based on Newton's second law, the Newton-Euler formulation explains how forces and moments induce motion. As demonstrated in [5], the equations of motion of an underwater vehicle can be written in the following form:

$$\underbrace{\mathbf{M}_{RB}\,\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\,\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta})}_{\text{rigid-body + hydrostatics}} + \underbrace{\mathbf{M}_A\,\dot{\boldsymbol{\nu}} + \mathbf{C}_A(\boldsymbol{\nu}_r)\,\boldsymbol{\nu}_r + \mathbf{D}(\boldsymbol{\nu}_r)\,\boldsymbol{\nu}_r}_{\text{hydrodynamics}} = \boldsymbol{\tau}. \tag{3.12}$$

where the system matrices and vectors are:

- $\mathbf{M}_{RB}$, $\mathbf{M}_A$: System inertia matrices (rigid-body and added mass)
- $\mathbf{C}_{RB}(\boldsymbol{\nu})$, $\mathbf{C}_A(\boldsymbol{\nu}_r)$: Coriolis and centripetal matrices (rigid-body and added mass Coriolis).
- $\mathbf{D}(\boldsymbol{\nu}_r) = \mathbf{D}_L + \mathbf{D}_Q(\boldsymbol{\nu}_r)$: Hydrodynamic damping matrix (linear and quadratic drag).
- $\mathbf{g}(\boldsymbol{\eta})$: Vector of gravitational and buoyancy forces and moments.
- $\boldsymbol{\nu}_r$: Velocity vector relative to the ocean current.

The equations of motion are divided into two parts. The first part contains the forces related to the rigid-body and hydrostatics, which are associated with $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$, while the second one is populated by forces related to the hydrodynamics, which are associated with the relative velocity vector $\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c$, with $\boldsymbol{\nu}_c$ the generalized ocean current velocity of an irrotational fluid, defined as

$$\boldsymbol{\nu}_c = (\boldsymbol{\nu}_c^b,\, 0,\, 0,\, 0) \quad \text{where} \quad \boldsymbol{\nu}_c^b = (u_c,\, v_c,\, w_c).$$

This model, considered as simulation (high-fidelity) model, represents a general formulation, effectively containing couplings between different degrees of freedom. In particular, the interactions between longitudinal and lateral motions introduce nonlinearities. Therefore, some standing assumptions have been adopted to simplify the model used for control design and synthesis purposes.

Control Model Assumptions.

Since the BlueROV2 has four thrusters for horizontal movement and two for vertical movement, the two directions can be treated separately for control purposes. Also, it is more practical to focus only on the horizontal dynamics for path-following purposes, since the movement mainly happens in that direction. Therefore, the analysis focuses on the three degrees of freedom that govern horizontal motion: *surge*, *sway*, and *yaw*.

Below, as adopted in [35], the assumptions made on the model are formalized.

Assumption 1: *Stabilized Heave, Roll and Pitch* $\Rightarrow \theta, \phi \approx 0$ and $w = 0$.

Due to the design of the AUV, roll and pitch can be neglected since its design naturally keeps these DOF stable. Additionally, because the vertical and horizontal motions are independent, heave control can also be neglected. This simplifies the system to be control to three degrees of freedom, resulting in:

$$\boldsymbol{\eta} := \begin{bmatrix} x & y & \psi \end{bmatrix}, \quad \boldsymbol{\nu} := \begin{bmatrix} u & v & r \end{bmatrix}, \quad \boldsymbol{\tau} := \begin{bmatrix} X & Y & N \end{bmatrix}.$$

Assumption 2: *Decoupled motion.*

The AUV operates at a constant or slowly varying forward velocity, so the total speed can be approximated by the surge velocity:

$$U \approx \sqrt{u^2 + v^2} \approx u.$$

This allows the model to be separated in a in a forward speed (surge) model and a sway–yaw subsystem for maneuvering (see Section 7.1.4 [5]).

Assumption 3: *Zero Current Velocity*, i.e., $\boldsymbol{v}_c = 0$.

The equation of motion (3.12) can be formulated as follows:

$$\mathbf{M}\,\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\,\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\,\boldsymbol{\nu} = \boldsymbol{\tau},$$

where the matrices are defined as:

$$\begin{aligned}
\mathbf{M} &= \mathbf{M}_{RB} + \mathbf{M}_A,\\
\mathbf{C}(\boldsymbol{\nu}) &= \mathbf{C}_{RB}(\boldsymbol{\nu}) + \mathbf{C}_A(\boldsymbol{\nu}),\\
\mathbf{D}(\boldsymbol{\nu}) &= \mathbf{D} + \mathbf{D}_n(\boldsymbol{\nu}).
\end{aligned}$$

Assumption 4: *Symmetry in the xz-plane.*

The AUV is assumed to be symmetric about the xz-plane (i.e., port-starboard symmetric), which means $I_{xy} = I_{yz} = 0$. This symmetry, along with aligning the body-fixed frame with the center of gravity, which means $\mathbf{r}_g = [0, 0, 0]^T$), reduces the rigid-body mass matrix to

$$\mathbf{M}_{RB} = \operatorname{diag}(m,\, m,\, I_z),$$

where m is the mass and I_z is the moment of inertia about the z-axis.

Also, recalling the rigid-body motion equation:

$$\mathbf{M}_{RB}\,\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\,\boldsymbol{\nu} = \boldsymbol{\tau}_{RB},$$

and, by considering a linear approximation around $u = U$ (constant), $v = 0$, and $r = 0$ (see Section 6.1 [5]), the equation simplifies to $\mathbf{M}_{RB}\,\dot{\boldsymbol{\nu}} + \mathbf{C}^*_{RB}\,\boldsymbol{\nu} = \boldsymbol{\tau}_{RB}$, with the rigid-body Coriolis matrix simplified as

$$\mathbf{C}^*_{RB} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & mU \\ 0 & 0 & mx_gU \end{bmatrix}.$$

Assumption 5: *Off-diagonal elements of the added mass matrix small compared to the diagonal.*

The off-diagonal elements in the added mass matrix are considered much smaller than the diagonal elements and can be neglected. Consequently, the added mass matrix simplifies to

$$\mathbf{M}_A = -\operatorname{diag}(X_{\dot{u}},\, Y_{\dot{v}},\, N_{\dot{r}}),$$

where the coefficients represent the added mass and inertia in surge, sway and yaw, respectively.

Assumption 6: *Coriolis modeled assuming $u \gg 0$ and 2nd order terms in v, w, r*

considered small.

The added Coriolis-centrifugal matrix can be written as:

$$\mathbf{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u \\ -Y_{\dot{v}}v - Y_{\dot{r}}r & X_{\dot{u}}u & 0 \end{bmatrix} \approx \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -X_{\dot{u}}u \\ 0 & X_{\dot{u}}u & 0 \end{bmatrix},$$

where only terms in the cruise speed remain, the term $-Y_{\dot{r}}r$ is neglected due to Assumption 5, and the term $-Y_{\dot{r}}v$ is neglected due to Assumption 2.

Assumption 7: *Noncoupled motion.*

Since the motion is non-coupled, it is assumed that the damping matrix $\mathbf{D}(\boldsymbol{\nu})$ has a diagonal structure (see Section 7.5.5 [5]). This matrix can be seen as composed of a linear and a non-linear (quadratic) part, where the nonlinear drag coefficients dominate at higher speeds. Thus, $\mathbf{D}(\boldsymbol{\nu})$ simplified as:

$$\begin{aligned} \mathbf{D}(\boldsymbol{\nu}) = &-\text{diag}(X_u,\, Y_v,\, N_r)+ \\ &- \text{diag}(X_{|u|u}\,|u|,\, Y_{|v|v}\,|v|,\, N_{|r|r}\,|r|). \end{aligned}$$

3.3 3D Maneuvering Model

Based on the previus assumptions, the nonlinear equation describing the motions of surge, sway and yaw of the AUV can be summarized as:

$$\begin{cases} \dot{\boldsymbol{\eta}} = \mathbf{J}_\Theta(\boldsymbol{\eta})\,\boldsymbol{\nu} \\ \dot{\boldsymbol{\nu}} = \dfrac{1}{\boldsymbol{M}}\left[-\mathbf{N}(\boldsymbol{\nu}_r) + \boldsymbol{\tau}(t)\right] \end{cases} \tag{3.13}$$

where:

$$\mathbf{J}_\Theta(\psi) \;=\; \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & 0 \\ 0 & 0 & I_z - N_{\dot{r}} \end{bmatrix},$$

$$\mathbf{N}(\boldsymbol{\nu}_r) := \mathbf{C}_{RB}(\boldsymbol{\nu}_r) + \mathbf{C}_A(\boldsymbol{\nu}_r) + \mathbf{D}_L + \mathbf{D}_Q(\boldsymbol{\nu}_r) =$$

$$= \begin{bmatrix} -X_u - X_{|u|u}|u_r| & -mr & 0 \\ mr & -Y_v & -X_{\dot{u}}u_r - Y_r \\ -Y_{\dot{v}}v_r & X_{\dot{u}}u_r - N_v & -N_r - N_{|r|r}|r| \end{bmatrix}.$$

3.3.1 Linearization of Maneuvering Model

A common approach for controller design is to linearize the nonlinear model around an operating point, thereby approximating the dynamics locally. The nonlinear kinetic model in Equation (6.8) can be written in terms of the vector function:

$$f(\boldsymbol{\nu}_r, \boldsymbol{\tau}) \;=\; \dot{\boldsymbol{\nu}}_r \;=\; -\,\mathbf{M}^{-1}\Big[\mathbf{N}(\boldsymbol{\nu}_r)\,\boldsymbol{\nu}_r\Big] \;+\; \mathbf{M}^{-1}\,\boldsymbol{\tau}, \quad \det(\mathbf{M}) \;\neq\; 0 \tag{3.14}$$

A linear approximation to the nonlinear system in Equation (3.14) can be obtained by choosing an operating point (ν_{r0}, τ_0) and then computing the linearization around that operating point.This means that the state variables, inputs and outputs becomes incremental and express the deviation around the the chosen operating point. By choosing a desired stationary state in terms of a cruise speed u_{r0}:

$$\boldsymbol{\nu}_{r0} = \Big[u_{r0},\, 0,\, 0\Big]^\top, \tag{3.15}$$

the corresponding stationary input τ_0 can be computed as:

$$\boldsymbol{\tau}_0 = \mathbf{C}(\boldsymbol{\nu}_{r0})\,\boldsymbol{\nu}_{r0} + \mathbf{D}(\boldsymbol{\nu}_{r0})\,\boldsymbol{\nu}_{r0} = -\begin{bmatrix} X_u + X_{|u|u}\,(u_{r0})\,u_{r0} \\ 0 \\ 0 \end{bmatrix}. \tag{3.16}$$

The linear deviation variables can now be defined as:

$$\Delta\boldsymbol{\nu}_r := \boldsymbol{\nu}_r - \boldsymbol{\nu}_{r0} = \Big[\Delta u_r,\, v_r,\, r_r\Big]^\top, \tag{3.17}$$

$$\Delta\boldsymbol{\tau} := \boldsymbol{\tau} - \boldsymbol{\tau}_0 = \Big[\Delta\tau_1,\, \tau_2,\, \tau_6\Big]^\top, \tag{3.18}$$

and a linear model in the state-space form can be written as:

$$\Delta\boldsymbol{\nu}_r = \mathbf{A}_\nu(u_{r0})\,\Delta\boldsymbol{\nu}_r + \mathbf{B}_\nu\,\Delta\boldsymbol{\tau}, \tag{3.19}$$

where $\mathbf{A}_\nu$ and $\mathbf{B}_\nu$ are obtained by computing the Jacobian matrices evaluating those at the operating point as follows:

$$\mathbf{A}_{\boldsymbol{\nu}}(u_{r0}) = \left.\frac{\partial\dot{\boldsymbol{\nu}}_r}{\partial\boldsymbol{\nu}_r}\right|_{\boldsymbol{\nu}_{r0},\,\boldsymbol{\tau}_0} = -\,\mathbf{M}^{-1}\left.\frac{\partial\Big(\mathbf{N}(\boldsymbol{\nu}_r)\,\boldsymbol{\nu}_r\Big)}{\partial\boldsymbol{\nu}_r}\right|_{\boldsymbol{\nu}_{r0},\,\boldsymbol{\tau}_0}, \tag{3.20}$$

$$\mathbf{B}_{\boldsymbol{\nu}} = \left.\frac{\partial \dot{\boldsymbol{\nu}}_r}{\partial \boldsymbol{\tau}}\right|_{\boldsymbol{\nu}_{r0},\, \boldsymbol{\tau}_0} = \mathbf{M}^{-1}. \tag{3.21}$$

Therefore, the forward speed and maneuvering linearized model are presented below.

3.3.2 Surge Dynamics

Under Assumption 2 and Assumption 4, the surge dynamics can be expressed as follows:

$$(m - X_{\dot{u}})\dot{u} - X_u u_r - X_{|u|u}|u_r|u_r = \tau_1, \tag{3.22}$$

where τ_1 represents the sum of control inputs and external forces acting in the surge direction.

3.3.3 Sway-Yaw Dynamics

The linearized maneuvering model can be formulated, following [5], as follows:

$$\mathbf{M}\,\dot{\boldsymbol{\nu}} + \mathbf{N}(u_o)\,\boldsymbol{\nu} = \boldsymbol{\tau}, \tag{3.23}$$

where $\boldsymbol{\nu} = [\boldsymbol{v},\, \boldsymbol{r}]^\top$ represents the sway and yaw velocities, $\boldsymbol{\tau}$ represents the force and moment applied by the thrusters and, for simplicity of notation,

$$\mathbf{N}(\boldsymbol{\nu}) = \mathbf{C}(\boldsymbol{\nu}) + \mathbf{D}(\boldsymbol{\nu}).$$

Based on Assumption 2, Assumption 3, and the assumption that the cruise speed is constant, i.e., $u = u_o \approx$ constant, the Coriolis and centripetal matrices for the sway and yaw model lead to:

$$\mathbf{C}(\boldsymbol{\nu})\,\boldsymbol{\nu} = \begin{bmatrix} 0 & (m - X_{\dot{u}})u_o \\ (X_{\dot{u}} - Y_{\dot{v}})u_o & m x_g u_o \end{bmatrix} \begin{bmatrix} v \\ r \end{bmatrix}.$$

The linear damping matrix in the sway-yaw plane is given by:

$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} -Y_v & -Y_r \\ -N_v & -N_r \end{bmatrix}, \tag{3.24}$$

while the system mass matrix can be written as

$$\mathbf{M} = \begin{bmatrix} m - Y_{\dot{v}} & m x_g - Y_{\dot{r}} \\ m x_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix}. \tag{3.25}$$

Rearranging the sway-yaw model for maneuvering recalling equation (3.26), the final expression remains as follows:

$$\begin{bmatrix} m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ mx_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} -Y_v & -Y_r \\ -N_v & -N_r \end{bmatrix} \begin{bmatrix} v \\ r \end{bmatrix} + \\ + \begin{bmatrix} 0 & (m - X_{\dot{u}})u_0 \\ (X_{\dot{u}} - Y_{\dot{v}})u_0 & mx_g u_0 \end{bmatrix} \begin{bmatrix} v \\ r \end{bmatrix} = \begin{bmatrix} \tau_2 \\ \tau_6 \end{bmatrix}. \tag{3.26}$$

In particular, assuming a constant cruise speed U, Equation (3.26) and Equation (3.22) can be combined as:

$$\begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{r} \end{bmatrix} + \\ + \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & (m - Y_{\dot{v}})U - Y_r \\ 0 & -N_v & (mx_g - Y_{\dot{r}})U - N_r \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_6 \end{bmatrix},$$

where surge is decoupled from the sway–yaw subsystem.

3.4 Thrust Configuration

The control force due to a thruster can be represented, assuming linearity, as:

$$\mathbf{F} = K\mathbf{u}, \tag{3.27}$$

where $\mathbf{u}$ is the control input and K is the thrust coefficient, which is used as a scaling factor mapping the control input to the thrust force. Since BlueROV2 has 6 thrusters, as shown in Figure 3.3, where the blue propellers rotate clockwise, the green propellers rotate counterclockwise and the red arrow indicates the positive surge direction, the thruster forces can be represented using the vector $\mathbf{F} = \begin{bmatrix} F_1 & F_2 & F_3 & F_4 & F_5 & F_6 \end{bmatrix}^T$, and the control inputs can be represented using the vector $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{bmatrix}^T$.

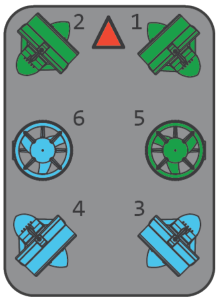

Figure 3.3: Layout of the thruster in BlueROV2. [36]

The forces and moments in 6 DOF, equal to the force vector $\mathbf{f} = [F_x, F_y, F_z]^\top$, can be expressed as:

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{f} \\ \mathbf{r} \times \mathbf{f} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ F_z l_y - F_y l_z \\ F_x l_z - F_z l_x \\ F_y l_x - F_x l_y \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix}, \tag{3.28}$$

where $\mathbf{r} = \begin{bmatrix} l_x & l_y & l_z \end{bmatrix}$ denote the moment arms.

The propulsion matrix K, here denoted as TAM, is used to distribute the force and moments in the desired degrees of freedom among the corresponding thrusters. Hence, considering the BlueROV2 with six thrusters, the generalized forces and moments $\boldsymbol{\tau} \in \mathbb{R}^6$, generated by the six thrusters in terms of the control inputs $\mathbf{u} \in \mathbb{R}^6$, can be modeled as:

$$\boldsymbol{\tau} = \text{TAM}(\alpha)\mathbf{u} \tag{3.29}$$

where TAM $\in \mathbb{R}^{6\times 6}$ and α denotes the angle at which each thruster is positioned relative to the AUV's forward direction [37].

3.5 BlueROV2 Parameters

The parameters used in this project are based on [38] and are summarized in Table 3.2. Furthermore, the TAM used in this project, as per equation 3.29, is given by:

$$\text{TAM} = \begin{bmatrix} 0.707 & 0.707 & -0.707 & -0.707 & 0 & 0 \\ -0.707 & 0.707 & -0.707 & 0.707 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0.051 & -0.051 & 0.051 & -0.051 & 0.111 & -0.111 \\ 0.051 & 0.051 & -0.051 & -0.051 & 0.002 & -0.002 \\ -0.167 & 0.167 & 0.175 & -0.175 & 0 & 0 \end{bmatrix}. \tag{3.30}$$

Parameters	**Symbol**	**Value**	**Unit**
Mass	m	11.5	kg
Center of gravity	$r_G = (x_g, y_g, z_g)$	$(0, 0, 0)$	m
Center of buoyancy	$r_B = (x_b, y_b, z_b)$	$(0, 0, 0.02)$	m
Inertia moment	$I = \text{diag}(I_x, I_y, I_z)$	$\text{diag}(0.16, 0.16, 0.16)$	$\text{kg}\,\text{m}^2$
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	$\text{kg}\,\text{m}^2/\text{rad}$
	$M_{\dot{q}}$	-0.12	$\text{kg}\,\text{m}^2/\text{rad}$
	$N_{\dot{r}}$	-0.12	$\text{kg}\,\text{m}^2/\text{rad}$
Linear Damping Parameters	X_u	-4.03	$\text{N}\,\text{s}/\text{m}$
	Y_v	-6.22	$\text{N}\,\text{s}/\text{m}$
	Z_w	-5.18	$\text{N}\,\text{s}/\text{m}$
	K_p	-0.07	$\text{N}\,\text{s}/\text{rad}$
	M_q	-0.07	$\text{N}\,\text{s}/\text{rad}$
	N_r	-0.07	$\text{N}\,\text{s}/\text{rad}$
Quadratic Damping Parameters	$X_{u\|u\|}$	-18.18	$\text{N}\,\text{s}^2/\text{m}^2$
	$Y_{v\|v\|}$	-21.66	$\text{N}\,\text{s}^2/\text{m}^2$
	$Z_{w\|w\|}$	-36.99	$\text{N}\,\text{s}^2/\text{m}^2$
	$K_{p\|p\|}$	-1.55	$\text{N}\,\text{s}^2/\text{rad}^2$
	$M_{q\|q\|}$	-1.55	$\text{N}\,\text{s}^2/\text{rad}^2$
	$N_{r\|r\|}$	-1.55	$\text{N}\,\text{s}^2/\text{rad}^2$

Table 3.2: BlueROV2 parameters.

Chapter 4

Path Planning

This following chapter outlines the first stage of the control architecture: path planning. To achieve this objective, an RRT* algorithm is implemented for this part. Section 4.1 discusses the logic and main principles of the RRT algorithm, while Section 4.2 shows the principles of RRT* and parameters selection.

Figure 4.1 illustrates the general workflow for motion control. Initially, the path is generated by using the target position identified by the active sonar. The RRT* algorithm takes this information as input and calculates a sequence of way-points. Here, the resulting sequence of way-points is transmitted to the Line-of-Sight (LOS) guidance module, which calculates the desired heading angle required for the AUV to follow the planned trajectory (see Chapter 5). This heading reference is then passed to a PID-based control system, where safety-critical constraints are applied through the use of Control Barrier Functions (CBFs - see Chapter 6). Finally, the optimized control inputs are managed by a control allocation system, i.e. the TAM in (3.30), which generates the thruster commands to the AUV.

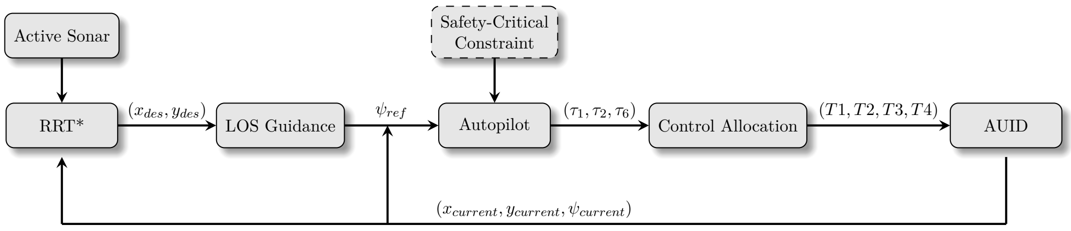

Figure 4.1: Schematic diagram for the closed-loop motion control of an AUV.

It is important to note that the active sonar shown in Figure 4.1 is used by the RRT* algorithm to determine the target's position for the trajectory planning. The choice of using an active sonar is motivated by the fact that other sensors, such as image-based sonars, are less suitable in this context, especially since the target might not be visible through simple cameras. By choosing an active (or similarly passive) sonar, the system ensures the capability of target detection without using any cameras or additional lighting, often used in underwater environment because of the absence of light. It is also worth mentioning that the specifications on how the sonar detects the target are beyond the scope of this thesis and, therefore, for this project, it is assumed that the target's position is provided by the sonar sensor. More detail on how the target's position is represented and identified is provided in Chapter 7 .

4.1 Dynamic Global Path Planning

The path planning problem consists on determining a path from a start point to a target goal while satisfying pre-specified constraints, such as by minimizing the length of the path while ensuring the collision avoidance. Here the environment, a wind farm, is known a priori, and it is considered static. Therefore, a global path planning algorithm can generate an optimal path. However, since the target is in motion, the generation of the path has to be dynamic. This leads to the implementation of a *global dynamic path planning* algorithm that can regenerate the path towards the moving target within the corresponding loop.

One common method adopted to achieve this objective is the Rapidly Exploring Random Tree (RRT) algorithm. This tehcnique works with a sampling-based approach, with a checking procedure on the collision that ensures the feasibility of the path. RRT constructs a tree by uniformly sampling the obstacle-free space and connecting these nodes until it reaches the target. A drawback of the RRT is the fact that often generates a suboptimal path and, for this reason, an improvement of this algorithm has been introduced in [12], i.e. the so-called RRT*. This algorithm improves upon RRT by incorporating a *rewiring* step. This step iteratively checks if a new sample node can lower the cost of an existing path. As more samples are added, it eventually converges to an optimal solution. To make sure that the moving target is reachable, some studies have been conducted, where the use of a RRT* algorithm has been adopted, such as [13]. This study shows that RRT* is able to manage large-scale optimization by exploring the full maneuvering space, and its solution improves as more samples are added, while keeping a reasonable computational effort. Therefore, the RRT* algorithm has been chosen to address the dynamic path planning task. It is important to remark that the project is developed in 2D, although studies on the performance of the same algorithm in a 3D and more complex environment can also be found, see e.g. [39].

4.1.1 Rapidly Exploring Random Tree (RRT)

Let $X \subset \mathbb{R}^d$ be the configuration space, $X_{\text{obs}} \subset X$ the obstacle region, $X_{\text{free}} = X \setminus X_{\text{obs}}$ the obstacle-free region, $x_{\text{start}} \in X_{\text{free}}$ the start state and $X_{\text{goal}} \subset X_{\text{free}}$ the goal region. The path planning problem consists of finding a sequence of way-points $\{x_1, x_2, \dots, x_n\} \subset X_{\text{free}}$ such that, when connected, these form a continuous path from x_{start} to some $x_{\text{goal}} \in X_{\text{goal}}$. RRT takes as input $(X_{\text{free}}, x_{\text{start}}, X_{\text{goal}})$, while the output that is produces is a graph $G = (V, E)$, where $V \subset X_{\text{free}}$ denotes the set of vertices, with cardinality $|V| \leq n + 1$, and $E \subseteq V \times V$ is the set of edges characterising G.

An algorithm, as cited by [12], is *asymptotically optimal* if, for a path planning problem whose best cost is c^*, the probability that the solution of the algorithm reaches c^* goes to 1 as the number of samples or iterations grows. That means, if the algorithm could run for a sufficiently long time, it will almost surely find a path with cost c^*. The standard RRT algorithm, as described in pseudocode within Algorithm 1, has been proved not to be asymptotically optimal [12]. In particular, it has been demonstrated that the best solution found by RRT never reaches the optimal value, but remains suboptimal with probability 1. Nonetheless, it has been shown to be *probabilistic complete* [40], which means that the probability of returning a failure solution tends to zero as the number of samples goes to infinity. Therefore, RRT is referred to as a *suboptimal* algorithm.

Algorithm 1 RRT

```
 1: $V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$
 2: for $i = 1, \ldots, n$ do
 3:     $x_{\text{rand}} \leftarrow \text{SampleFree}_i$
 4:     $x_{\text{nearest}} \leftarrow \text{Nearest}(G = (V, E), x_{\text{rand}})$
 5:     $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{rand}})$
 6:     if ObstacleFree($x_{\text{nearest}}, x_{\text{new}}$) then
 7:         $V \leftarrow V \cup \{x_{\text{new}}\}$
 8:         $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$
 9:     end if
10: end for
11: return $G = (V, E)$
```

The RRT algorihtm is built as an iterative procedure that incrementally builds a tree starting from an initial state and extends it randomly in the configuration space. At each iteration, a random node $x_{\text{rand}} \in X_{\text{free}}$ is sampled and the closest node x_{nearest} is identified (using an Eucliden norm in a 2D space, in this case). A steering function then generates a new node x_{new} that is present along the segment between x_{rand} and x_{nearest}. If the edge between x_{nearest} and x_{new} is completely outside X_{obs}, both x_{new} and the edge are added to the tree, otherwise the algorithm continues with the following iteration. The process terminates either when the number of iterations has reached the limit or when a path is effectively found. The pseudocode of this process is shown in Algorithm 1, based on [12].

4.2 RRT Star (RRT*)

To overcome the problem of asymptotic optimality described within this section, RRT* has been introduced in [12], and proven to be asymptotically optimal. This has been performed by associating a cost with each path and by adding a rewiring step to recalculate the optimal solution tree. With these modifications, RRT* results in an asymptotically optimal algorithm, in the sense of [12].

Algorithm 2 RRT*

```
 1: $V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$
 2: for $i = 1, \ldots, n$ do
 3:     $x_{rand} \leftarrow \text{SampleFree}_i$
 4:     $x_{nearest} \leftarrow \text{Nearest}(G = (V, E), x_{rand})$
 5:     $x_{new} \leftarrow \text{Steer}(x_{nearest}, x_{rand})$
 6:     if ObstacleFree($x_{nearest}, x_{new}$) then
 7:         $X_{near} \leftarrow \text{Near}(G = (V, E), x_{new}, r(\text{card}(V)))$
 8:         $V \leftarrow V \cup \{x_{new}\}$
 9:         $x_{min} \leftarrow x_{nearest}$; $c_{min} \leftarrow \text{Cost}(x_{nearest}) + c(\text{Line}(x_{nearest}, x_{new}))$
10:         for $x_{near} \in X_{near}$ do                ▷ Connect along a minimum-cost path
11:             if CollisionFree($x_{near}, x_{new}$) $\wedge$ $\text{Cost}(x_{near}) + c(\text{Line}(x_{near}, x_{new})) <$
    $c_{min}$ then
12:                 $x_{min} \leftarrow x_{near}$; $c_{min} \leftarrow \text{Cost}(x_{near}) + c(\text{Line}(x_{near}, x_{new}))$
13:             end if
14:         end for
15:         $E \leftarrow E \cup \{(x_{min}, x_{new})\}$
16:         for $x_{near} \in X_{near}$ do                ▷ Rewire the tree
17:             if CollisionFree($x_{new}, x_{near}$) $\wedge$ $\text{Cost}(x_{new}) + c(\text{Line}(x_{new}, x_{near})) <$
    $\text{Cost}(x_{near})$ then
18:                 $x_{parent} \leftarrow \text{Parent}(x_{near})$
19:                 $E \leftarrow (E \setminus \{(x_{parent}, x_{near})\}) \cup \{(x_{new}, x_{near})\}$
20:             end if
21:         end for
22:     end if
23: end for
24: return $G = (V, E)$
```

RRT* follows the same structure as RRT up until the steering function produces a new node x_{new}. At that point, the subset X_{near} is introduced in RRT*, where X_{near} is the set of nodes in $V \subset X_{\text{free}}$ that are inside a specified radius r_n of x_{new}. Which means, $X_{\text{near}} = \{\, x \in V : d(x,\, x_{\text{new}}) \leq r_n \}$, where $d(\cdot,\cdot)$ is the Euclidean norm. Therefore, once x_{new} is sampled, all nodes in X_{near} are grouped for a potential rewiring and cost updates.

x_{new} is then connected to an $x_{\text{near}} \in X_{\text{near}}$ which minimizes the cost of the path. After this connection, all other node in X_{near} is checked to see if changing their parent to x_{new} would reduce their cost. If the new cost is lower than the current one, the rewiring loop updates the parent node from x_{parent} to x_{new} and modify the edge $(x_{\text{parent}}, x_{\text{near}})$ with $(x_{\text{new}}, x_{\text{near}})$. The pseudocode is presented in Algorithm 2.

To form the set X_{near}, the algorithm uses a function, `Near`, which returns all nodes within a certain radius of x_{new}. This radius is defined as the minimum between a variable radius $r(\text{card}(V))$ and a fixed constant ε. The variable radius is given by

$$r(\text{card}(V)) = \min \left\{ \gamma_{\text{RRT}^*} \left(\frac{\log(\text{card}(V))}{\text{card}(V)} \right)^{\frac{1}{d}}, \varepsilon \right\}, \tag{4.1}$$

where γ_{RRT^*} is a positive constant that depends on X_{free} and the overall configuration space Q, d is the dimension of X, card(V) is the number of the total vertices of the optimal path and ε is a positive constants.

4.3 Path Generation based on Way-Points

After the generation of the sequence of way-points, the path is created by using straight lines and circle arcs to connect the way-points, as shown in Figure 4.2. This result is originally due to Dubins [41] and formulated by [5] as:

The shortest path (minimum time) between two configurations (x, y, ψ) of a craft moving at constant speed U is a path formed by straight lines and circular arc segments.

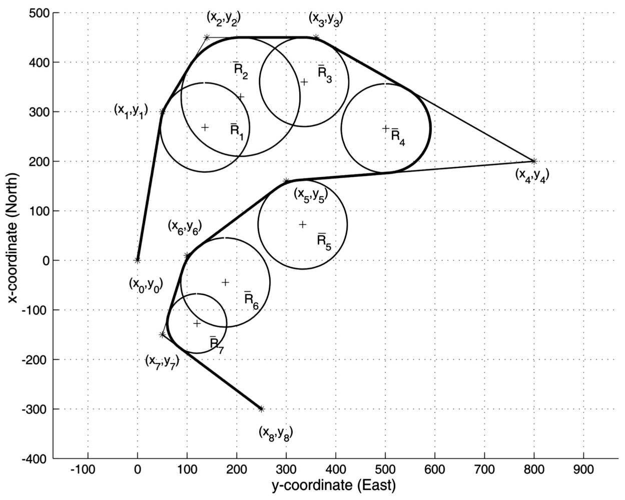

Figure 4.2: Path based on straight-lines and circular arcs [5].

This strategy is chosen because of its simplicity, while still keeping a good efficiency compared to e.g. a cubic interpolation. With this strategy, around each way-point a circle with radius $\bar{R}_i$ is defined, as shown in Figure 4.2. The point where this circle intersects the line is the turning point of the AUV. Therefore, the radius of the circle that is inside three following way-point can be computed from R_i as

$$\bar{R}_i = R_i \tan(\alpha_i), \quad i = 1, \ldots, n, \tag{4.2}$$

where α_i is defined in Figure 4.3 as α_1 and R_i is the radius of acceptance to guarantee a smooth transition between two segments of the path. Therefore, to ensure a feasible transition, three consecutive way-points must be selected such that:

$$\bar{R}_i \geq R_{\mathrm{min}}, \tag{4.3}$$

where R_{min} is the minimum turning radius, such that three consecutive way-points can not form an angle too tight that would create problems to the autopilot subsystem.

To sum up, in order to guarantee a smooth path following process, the parameters to take into account are:

- **Acceptance radius** R_i: The area where the AUV detects a way-point and starts to turn.
- **Turning radius** $\bar{R}_i$: The minimum curvature radius between two segments of the path.
- **Turn angle** α_i: The angle between two segments of the path.

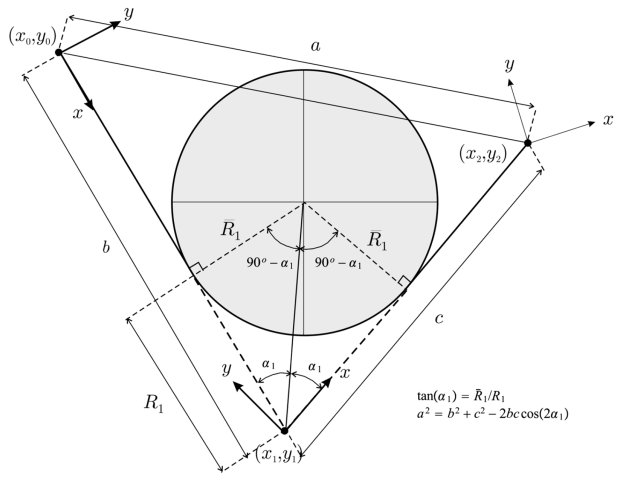

Figure 4.3: Circle inscribed between three consecutive way-points.

Therefore, the AUV operates in a safe zone if there is a check on the turning radius $\bar{R}_i$ (which means on the α_i), that has to be always greater than the minimum turning radius R_{min}. A similar approach has been examined in [42], where an angle factor constraint is incorporated into the Smooth-RRT algorithm to constraint the exploration process and optimize the planned path.

4.3.1 Parameters in RRT*

The performance of the RRT* algorithm depends on the choice of the following parameters, whose values depend on the specific problem and on the environment effectively being considered. The main parameters are:

- **Length Step (step_len)**: Length of the step used for the expansion of the tree.
- **Goal Sample Rate (goal_sample_rate)**: Probability of sampling a new node near the goal.
- **Search Radius (search_radius)**: Search radius used for the rewiring (optimization) of the tree.
- **Maximum Iteration (iter_max)**: Maximum number of iterations.
- **Alpha (alpha_1)**: Minimum turning angle between two segments.

In particular:

- **step_len**: This parameter plays a crucial role in the tree expansion process of Algorithm 2. Specifically, it is applied in `Steer` (line 5), where a new node is generated by moving from the nearest node (x_{nearest}) to the random sampled one (x_{rand}). The distance covered in this step is limited by `step_len`, ensuring that the tree grows incrementally and to avoid connections that are too far, avoiding encountering obstacles during the path. For the specific simulation, a wind farm of 20×20 meters has been defined. In order to have a good exploration and efficiency, the `step_len` parameter has been tuned to 0.8 meter.

- **goal_sample_rate**: During the sampling phase of a new node (line 3 of Algorithm 2), the algorithm chooses it by randomly picking in a uniform distribution between 0 and 1. If this value is greater than `goal_sample_rate`, the new node is generated randomly along the distance of `step_len` from the previous node. Otherwise, the goal node is chosen. Therefore, `goal_sample_rate` represents the probability of directly sampling the goal instead of a random node. This value is chosen to be 0.8, and has been selected by taking into account the specific environment (more detail on the environment in Chapter 7), which is not completely covered of obstacles. Indeed, in a windfarm, it can be say that the AUV should not expect to encounter many obstacles given the relatively large dimensions of the wind turbine compared to the vehicle. Therefore, using a high `goal_sample_rate` value, the search is more directed towards the goal, avoiding the exploration of unnecessary areas.

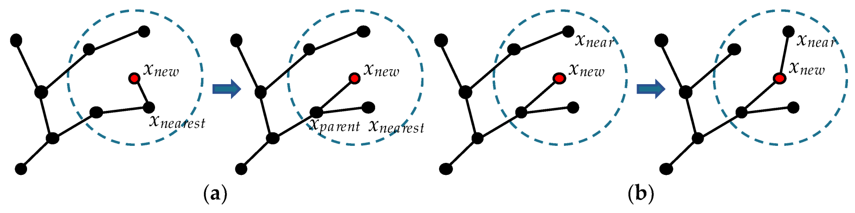

Figure 4.4: (a) ChooseParent phase(RRT*), (b) Rewire phase (RRT*). [43] Line 18-19 of Algorithm 2.

- **search_radius**: In practice, when a new node x_{new} is created, the function `Near` (line 7) selects all existing nodes within the radius defined by this parameter. This value is expressed as r in the pseudo code, which is nothing but the static value ε of Equation 4.1. It is important to note that a fixed value has been chosen for this parameter rather than the dynamic approach described. These nodes are then evaluated to determine if connecting them to x_{new} could lead to a lower cost of the path. A high `search_radius` value increases the chance of finding better paths but slows down the algorithm because more nodes must be checked, on the contrary, a smaller value speeds up the process but may find a suboptimal path. Setting it to 4.0 is an effective compromise between exploration and computational performance.

- **iter_max**: This value is the maximum number of iterations that the algorithm computes in order to deliver the optimized path. This value is expressed in the for loop of the Algorithm 2 in line 2. After a tuning phase, a value of 4000 iterations has been set.

- **alpha_1**: The constraint on the minimum turning angle is added in the following main functions of the Algorithm 2: `Steer` (line 5) and during the `Rewire the tree` phase in line 16. In particular, in the `Steer` function, a new node is not considered if the angle formed by the segment from its parent's parent to its parent and the segment from its parent to the new node is greater than or equal to $2\alpha_i$. Similarly, this check has been added in line 17 as an addition check after the `CollisionFree` and `Cost` functions. A value of 90 deg is set to this parameter. The phases of rewiring and choosing a new parent are shown in Figure 4.4.

Finally, Table 4.1 gives an overview of the values of the algorithm's parameters. Note that α_1 is express in rad in the Table. The results of the choice of these parameters are presented in Chapter 7.

Parameter	**Value**
`step_len`	0.8
`goal_sample_rate`	0.8
`search_radius`	4.0
`iter_max`	4000
`alpha1`	0.78

Table 4.1: Summary of the RRT* parameters.

Chapter 5

Guidance

The guidance module represents the second component of the control architecture shown in Figure 4.1. The way-points generated by the path planning module, which define the desired trajectory, are directly used as in ”input” to the content presented within this subsection. The objective herein is to compute a reference yaw angle that enables the AUV to follow the calculated path. In the following section, this problem is addressed using a LOS (Line-of-Sight) guidance algorithm.

5.1 Line-Of-Sight Guidance

In marine guidance, the LOS is the line that connects the vessel to a target point. The target point, $P(x_{\text{LOS}}, y_{\text{LOS}})$, is the point tangent to the path at a fixed lookahead distance $\Delta h > 0$ ahead of the AUV's current position $P(x, y)$ projected on the path, as shown in Figure 5.2.

As studied in [44], the 3D path following problem can be addressed by decoupling the problem in horizontal and vertical components. The horizontal guidance has the objective of generating heading reference trajectories to make sure that the vehicle converges to a straight line on the xy-plane. As already mentioned, this project focuses only on the horizontal plane. In particular, as illustrated in Figure 5.1, in the previous phase, the RRT* block generates the waypoints (x_k, y_k), which are processed to compute the desired heading angle, ψ_d, to be then provided to the autopilot. This phase also involves the knowledge of the current state x and y of the AUV.

5.1.1 Horizontal LOS Guidance Law

The kinematic equation of the AUV can be expressed as:

$$\begin{aligned}
\dot{x} &= u\cos(\psi) - v\sin(\psi) \approx U\cos(\psi),\\
\dot{y} &= u\sin(\psi) + v\cos(\psi) \approx U\sin(\psi),\\
\dot{\psi} &= r,
\end{aligned}$$

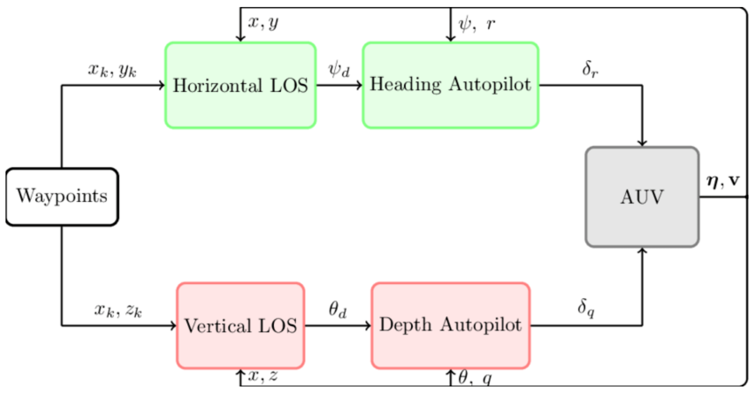

Figure 5.1: Decoupled LOS guidance for the horizontal and vertical planes ([44], p 10).

where the following approximation

$$U = \sqrt{u^2 + v^2} \approx u$$

is adopted based on Assumption 2 (see Section 3.3). An additional assumption is adopted with respect to the velocity, i.e.

$$0 < U_{\min} \leq U \leq U_{\max},$$

required to guarantee stability of the LOS guidance (see [44]).

The objective of the path following in the horizontal plane consists in following the line connecting the waypoints (x_k, y_k) and (x_{k+1}, y_{k+1}). The *along-track distance* and the *cross-track error* of a vehicle in position (x, y) are given by:

$$\begin{bmatrix} x_e \\ y_e \end{bmatrix} = \mathbf{R}^\top(\gamma_p) \begin{bmatrix} x - x_k \\ y - y_k \end{bmatrix},$$

where (x_k, y_k) is the position of the k-th waypoint expressed in the NED frame, and the rotation matrix from the inertial to the path reference frame is given by:

$$\mathbf{R}(\gamma_p) = \begin{bmatrix} \cos(\gamma_p) & -\sin(\gamma_p) \\ \sin(\gamma_p) & \cos(\gamma_p) \end{bmatrix} \in SO(2),$$

with γ_p the horizontal path-tangential angle:

$$\gamma_p = \text{atan2}(y_{k+1} - y_k, x_{k+1} - x_k).$$

In particular,

$$\begin{aligned} x_e &= \ (x - x_k)\cos(\gamma_p) + (y - y_k)\sin(\gamma_p), \\ y_e &= -(x - x_k)\sin(\gamma_p) + (y - y_k)\cos(\gamma_p). \end{aligned}$$

In order to solve a path-following problem, only the *cross-track error* is relevant since $y(t) = 0$ means that the vehicle has converged to the straight line. Therefore, it is sufficient to guarantee that:

$$\lim_{t \to \infty} y_e(t) = 0.$$

Two different guidance strategies can be used in order to have $y(t) = 0$: *Enclosure-based steering* and *Lookahead-based steering.* The second one has been selected as it is more suitable for marine applications (see the arguments in [5]).

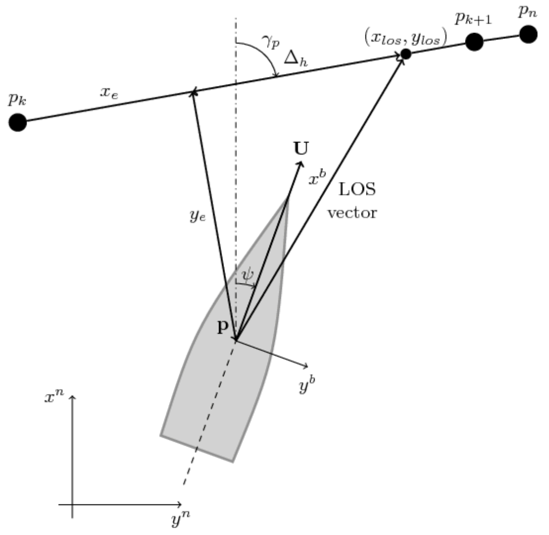

Figure 5.2: LOS guidance for the horizontal plane [5].

The lookahead-based guidance law, as illustrated in Figure 5.2, is given by (see [16]):

$$\psi_d = \gamma_p + \arctan\left(\frac{-y_e}{\Delta_h}\right),$$

where γ_p is as (5.1.1), y_e is the cross-track error and Δ_h is the lookahead distance. Note that the waypoints defined as (x_k, y_k) and (x_{k+1}, y_{k+1}), in the Figure 5.2 are denoted as p_k and p_{k+1}, however they correspond to the same points.

Note that in the presence of disturbances or during turns, the heading angle ψ_d and the course angle χ_d are not aligned anymore and are insted related in the following way:

$$\chi_d = \psi_d + \beta,$$

and therefore the desired heading angle is:

$$\psi_d = \gamma_p + \arctan\left(\frac{-y_e}{\Delta_h}\right) - \beta,$$

where

$$\beta = \text{atan2}(v, u).$$

However, based on Assumption 2 (see Section 3.3), this term can be neglected.

5.2 Parameters in LOS Guidance

As explained in Section 4.3, if the path is composed by n line segments connected by $n+1$ waypoints, a switching law is needed in order to know when the following point needs to be considered.

The mechanism of the *circle of acceptance* is employed. In this way, the next waypoint, (x_{k+1}, y_{k+1}), can be selected if the AUV is inside the region of the circle defined by the radius R centered at the waypoint. The formulation can be expressed as:

$$[x_{k+1} - x(t)]^2 + [y_{k+1} - y(t)]^2 \leq R^2.$$

If this relation is respected, the AUV can switch to take into account the next waypoint (x_{k+1}, y_{k+1}).

Therefore, the main parameters of the LOS guidance are:

- **Radius of Acceptance (radius_of_acceptance)**: Area around a waypoint where there is the transition to the next waypoint.
- **Lookahead Distance (lookahead_distance)**: How far ahead the LOS waypoint is chosen to be considered for planning the path.

In particular:

- **radius_of_acceptance**: A fixed value of 0.4 has been chosen for this parameter. This value is selected since the step size of the RRT* has been set to 1, and considering the length of the AUV, which is 0.5 m, this choice is considered reasonable.
- **lookahead_distance**: A value of 0.8 is chosen for this parameter, based on [5], which states that this parameter (specified in meters) usually takes values between 1.5 and 2.5 of the vehicle length. This choice is considered reasonable since a smaller value would make navigation more irregular, while a larger value would guarantee that the AUV reaches each waypoint.

The parameters for the LOS are summarised in Table 5.1.The results of the choice of these parameters are presented in detail within Chapter 7.

Parameter	**Value**
`radius_of_acceptance`	0.4
`lookahead_distance`	0.8

Table 5.1: Summary of LOS parameters.

Chapter 6

Control System

After generating the path as a sequence of waypoints from the RRT* module and implementing a LOS guidance system, which provides the references to follow the trajectory, the module in charge of tracking these references is the *Control System*, as shown in Figure 4.1.

In the following sections, first, a model-based autopilot design is introduced in Section 6.1, in particular, the control on the speed, detailed in Subsection 6.1.3, and the control on the steering, in Subsection 6.1.2, are shown. Afterwards, to ensure robust lane-keeping performance, even in the presence of sudden disturbances like currents, an approach based on CBFs is introduced in Section 6.2.

6.1 Autopilot

An autopilot is a system that controls a vehicle without the intervention of manual steering. In 1922, Nicholas Minorsky published his study on how a pilot steered a ship and developed what is now called PID control [45]. The first autopilots implemented used this approach.

More recently, systems based on Linear Quadratic Gaussian (LQG) control and H_∞ control have replaced these methods, mostly because these are efficient in filtering and attenuating first-order wave forces [46]. The Kalman filter and Linear Quadratic (LQ) control enabled LQG control, which is useful for purposes of Multi-Input Multi-Output (MIMO) ship control. Using a Kalman filter has the advantage of estimating wave-frequency motions and handling the coupling between surge, sway, and yaw, a task that three separate PID controllers cannot accomplish. For more details, see [5, Chapter 11].

However, the PID-based approach can yield good practical results, as evidenced in [47]. In this study, a separate controller is developed for each of the three main subsystems: a steering autopilot to manage heading errors, a diving system to stabilize the depth and a speed system for the surge motion. It is assumed that the controllers do not interact (or only interact lightly) with each other. Experiments confirmed this, showing that the roll and pitch angles stayed small enough to keep the system stable and that these worked independently.

Dividing the control task into three independent processes simplifies the design and tuning processes while still achieving stability and good control. By treating heading, depth and speed independently, uncertainties in the hydrodynamic model and external disturbances like waves are relaxed. Motivated by this, a PID-based approach was chosen for the two uncoupled subsystems of the speed and steering control.

6.1.1 PID Pole Placement

Since both the surge and yaw systems can be modeled as first-order systems, as shown in Equation (3.23) and Equation (3.22), a linear PID approach is adopted for control. In this configuration, the closed-loop system behaves as a second-order system. The following presents a pole placement approach for calculating the control parameters. These parameters are subsequently applied for surge and steer control.

A mass-damper-spring system is used to demonstrate the main concept. The two following equations represent the same system equivalently:

$$m\ddot{x} + d\dot{x} + kx = 0, \tag{6.1}$$

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = 0, \tag{6.2}$$

implying that:

$$2\zeta\omega_n = \frac{d}{m}, \quad \omega_n^2 = \frac{k}{m}$$

Therefore, for second-order systems, it is convenient to define:

$$\text{Natural frequency:} \quad \omega_n = \sqrt{\frac{k}{m}},$$

$$\text{Relative damping ratio:} \quad \zeta = \frac{d}{2m\omega_n}.$$

The control law based on a PID control can be formulated as:

$$\tau = K_p\tilde{x} + K_d\dot{\tilde{x}} + K_i\int_0^t \tilde{x}(\tau)d\tau$$

where $K_p > 0$, $K_d > 0$, and $K_i > 0$, and the tracking error is defined as $\tilde{x} = x - x_d$, where x is the state of the system and x_d is the desired state. After algebraic calculations, the natural frequency and relative damping ratio become as:

$$\omega_n = \sqrt{\frac{k + K_p}{m + K_m}}, \quad \zeta = \frac{d + K_d}{2(m + K_m)\omega_n}.$$

Specifying a desired ω_n and ζ, the values of K_p and K_d can be chosen as:

$$K_p = (m + K_m)\omega_n^2 - k, \tag{6.3}$$

$$K_d = 2\zeta\omega_n(m + K_m) - d, \tag{6.4}$$

and a *rule-of-thumb* imposes that

$$K_i = \frac{\omega_n}{10} K_p. \tag{6.5}$$

This approach has been adopted to estimate the gain of the PID for the steering and surge control as shown in the next sections.

6.1.2 Steering Control Autopilot

Steering control is the process of regulating the yaw dynamics of a vehicle in order to maintain a desired heading. To accomplish this, the Nomoto model [48] is often used due to its simplicity and effectiveness. This model is widely employed in autopilot design. This model can be derived from the linearized maneuvering model as shown below.

Recalling the manouvering model (3.23):

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{N}(u_o)\,\boldsymbol{\nu} = \boldsymbol{\tau},$$

or, in expanded form:

$$\begin{bmatrix} m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ mx_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} -Y_v & -Y_r \\ -N_v & -N_r \end{bmatrix} \begin{bmatrix} v \\ r \end{bmatrix} + \\ + \begin{bmatrix} 0 & (m - X_{\dot{u}})\,u_o \\ (X_{\dot{u}} - Y_{\dot{v}})\,u_o & mx_g\,u_o \end{bmatrix} \begin{bmatrix} v \\ r \end{bmatrix} = \begin{bmatrix} \tau_2 \\ \tau_6 \end{bmatrix}. \tag{6.6}$$

in order to control the yaw rate, r is selected as output:

$$r = c^T v, \quad c^T = [0,\, 1].$$

The Laplace transform yields to:

$$\frac{r}{\tau_6}(s) = \frac{K(1 + T_3 s)}{(1 + T_1 s)(1 + T_2 s)}, \tag{6.7}$$

where T_1, T_2, T_3, and K are constants values explained more in detail below.

Equation (6.7) is referred to as *Nomoto's second-order model.*

For practical reason, the *first-order Nomoto model* is used in this project. It is obtained by defining the equivalent time constant:

$$T := T_1 + T_2 - T_3,$$

such that

$$\frac{r}{\tau_6}(s) = \frac{K}{1+Ts}.$$

Finally, since $\dot{\psi} = r$, it can also be written as:

$$\frac{\psi}{\tau_6}(s) = \frac{K}{s(1+Ts)}.$$

This last relation is the transfer function used in most commercial autopilot systems.

In the time domain, the Nomoto's first-order model can be expressed as:

$$T\ddot{\psi} + \dot{\psi} = K\,\tau_6$$

where ψ is the yaw angle and τ_6 is the control input. Recalling Equation (6.1), the parameters here under consideration are given by:

$$m = \frac{T}{K}, \quad d = \frac{1}{K}, \quad k = 0,$$

where

$$T = T_1 + T_2 - T_3 = -\frac{I_z - N_{\dot{r}}}{N_r},$$

$$K = -\frac{1}{N_r}.$$

Given these parameters and recalling Equations (6.3), (6.4), and (6.5), the gains are finally defined as:

$$K_p = \frac{\omega_n^2 T}{K} > 0,$$

$$K_d = \frac{2\zeta\omega_n T - 1}{K} > 0,$$

$$K_i = \frac{\omega_n^3 T}{10K} > 0.$$

Recalling that the corresponding values are provided in Section 3.5, the obtained values for the control gains are $K_p = 25$, $K_i = 0$, and $K_d = 8$.

6.1.3 Speed Control Autopilot

Similarly, recalling the surge model described in Equation (3.22)

$$(m - X_{\dot{u}})\dot{u} - X_u u - X_{|u|u}|u|u = \tau_1,$$

according to [5], this can be simplified as:

$$(m - X_{\dot{u}})\dot{u} - X_u u = \tau_1 + T_{\text{loss}},$$

where T_{loss} represents coupling terms and environmental disturbances. This is a first-order system which can be controlled using a PI-controller of the form:

$$\tau_1 = -K_p(u - u_d) - K_i \int_0^t (u - u_d) d\tau.$$

Therefore, the error dynamics is:

$$(m - X_{\dot{u}})\dot{z} + (K_p - X_u)z + K_i \int_0^t z d\tau = T_{\text{loss}},$$

where:

$$z = \int_0^t (u - u_d) d\tau,$$

with u representing the measured surge velocity and u_d denotes the desired surge velocity. In order to guarantee a closed-loop stability, a pole placement algorithm is chosen and the gains $K_p > 0$ and $K_i > 0$ are such that:

$$\frac{K_p - X_u}{m - X_{\dot{u}}} = 2\zeta\omega_n,$$

$$\frac{K_i}{m - X_{\dot{u}}} = \omega_n^2,$$

where the relative damping ratio ζ and natural frequency ω_n are chosen according to the desired behavior. The determined control gains are $K_p = 7$, $K_i = 2$, and $K_d = 0$.

Simulation results of both the speed and steering control can be seen in Section 7.2.

6.2 Safety-Critical Control

The control framework now presented is a type of control that is able to ensure both performance objectives and safety constraints, an important feature that an autonomous system must satisfy. As highlighted in [20], to synthesise and compute this type of control system, the use of a Quadratic Program (QP) is often considered. In this framework, a control objective is defined by a Control Lyapunov Function (CLF) and a safe region is defined by a Control Barrier Function (CBF). The QP is implemented with the objective of keeping the system stable while making sure it stays in a safe region. Here, the stabilization objective is considered as a soft constraint through a relaxation factor, while safety is considered as a hard constraint.

This framework can be applied to any kind of controller, not just those based on CLFs. In other words, any control law that guarantees stability can be modified, as little as possible, to ensure safety. This can be done by solving an optimization problem that finds the smallest adjustment needed. The modified (optimized) controller will remain as close as possible to the original (nominal) one while satisfying the safety requirements.

As already mentioned, in a marine environment, unexpected currents can occur. These currents may cause the AUV to potentially leave the planned trajectory. To overcome this problem, a safety controller that enforces a maximum distance from the trajectory was implemented. This safety mechanism is realized through a CBF constraint. It behaves as a ”barrier” in the sense that if the AUV reaches a maximum distance from its predefined trajectory, the barrier activates and prevents the vehicle from going beyond these limits. The selection of the barrier is a critical aspect since it has to satisfy specific criteria. The following sections provide some background and notation for CBFs, showing its characteristics.

6.2.1 Control Barrier Function

Control Barrier Functions effectively act as “barriers”, preventing a system’s state from migrating into unsafe regions, thereby maintaining safety at all times. To define a CBF, it is first needed to specify a safe set $C = \{x \in \mathbb{R}^n : h(x) \geq 0\}$, which represents the region of the state space in which the system must remain. The CBF is then derived from this safe set in such a way that any control input that would drive the system toward an unsafe region is “corrected” to keep the system within C. Practically, the CBF integrates safety constraints directly into the control design, ensuring that every control input generated maintains the system within safe limits. In control-affine systems, these constraints can be incorporated into a quadratic program (QP) formulation, ensuring that the resulting control actions both respect the safety requirements and achieve the system’s performance goals. In this section, the fundamental aspects of control barrier functions are presented, such as the notions of safety, safe sets, and a method to ensure safety with minimal intervention in a nominal control law.

A nonlinear affine control system is considered, of the form:

$$\dot{x} = f(x) + g(x)u, \tag{6.8}$$

where f and g are locally Lipschitz, $x \in D \subset \mathbb{R}^n$, and $u \in U \subset \mathbb{R}^m$ is the set of admissible inputs.

As already mentioned, ensuring both stability and safety is a fundamental requirement for an autonomous system. While stability focuses on guiding the system to a specific state, safety ensures that it remains within a predefined region, known as the *safe set*. To formalize this last concept, we define the safe set $\mathcal{C}$ as the *superlevel set* of a continuously differentiable function $h : D \subset \mathbb{R}^n \to \mathbb{R}$. This means that the system operates within the region where $h(x)$ is non-negative, preventing it from entering unsafe states, yielding:

$$\begin{aligned} \mathcal{C} &= \{x \in D \subset \mathbb{R}^n : h(x) \geq 0\}, \\ \partial\mathcal{C} &= \{x \in D \subset \mathbb{R}^n : h(x) = 0\}, \\ \mathrm{Int}(\mathcal{C}) &= \{x \in D \subset \mathbb{R}^n : h(x) > 0\}. \end{aligned} \tag{6.9}$$

$\mathcal{C}$ is referred as the *safe set*. The concept of safety in a control system can be explained as follows.

Let $u = k(x)$ be a feedback controller such that the resulting dynamical system

$$\dot{x} = f_{\text{cl}}(x) := f(x) + g(x)k(x), \tag{6.10}$$

is *locally Lipschitz*. A function f, defined in $f : D \subseteq \mathbb{R}^n \to \mathbb{R}^m$, is *locally Lipschitz* at a point $x_0 \in D$ if there exists an open neighborhood $U \subseteq D$ of x_0 and a constant $L > 0$ such that, for all $x, y \in U$,

$$\|f(x) - f(y)\| \leq L\,\|x - y\|.$$

If this property holds for every $x_0 \in D$, then f is called locally Lipschitz on D. Due to this assumption, for any initial condition $x_0 \in D$, there exists a maximum interval of existence $I(x_0) = [0, \tau_{\max})$ such that $x(t)$ is the unique solution to (6.10) on $I(x_0)$.

Therefore, this allows to define the concept of *forward invariance* and *safety* [20]:

Definition 1. (*Safety*). *The set $\mathcal{C}$ is forward invariant if for every $x_0 \in \mathcal{C}$, $x(t) \in \mathcal{C}$ for $x(0) = x_0$ and all $t \in I(x_0)$. The system* (6.10) *is safe with respect to the set $\mathcal{C}$ if the set $\mathcal{C}$ is forward invariant.*

A commonly used barrier function is:

$$B(x) = -\log\left(\frac{h(x)}{1 + h(x)}\right). \tag{6.11}$$

To guarantee that $\text{Int}(\mathcal{C})$ remains forward invariant, a strict condition would be to impose:

$$\dot{B} \leq 0.$$

However, this condition can be too restrictive, as it requires all sublevel sets of B to be invariant. Therefore, a relaxed factor is introduced, leading to:

$$\dot{B} \leq \frac{\gamma}{B}, \tag{6.12}$$

where γ is positive. This modification allows $\dot{B}$ to grow when solutions are far from the boundary of C, while ensuring that when they get closer to the boundary, the rate of growth decreases to zero.

Therefore, based on Definition 1 and the nonlinear affine control system in (6.8), a Control Barrier Function is defined as follows:

Definition 2. (*CBF*) Let $\mathcal{C} \subset D \subset \mathbb{R}^n$ be the superlevel set of a continuously differentiable function $h : D \to \mathbb{R}$, then h is a Control Barrier Function (CBF) if there exists an extended class $\mathcal{K}_\infty$ function α such that for the control system (6.8):

$$\sup_{u \in \mathcal{U}} \dot{h}(x, u) \geq -\alpha(h(x)), \tag{6.13}$$

for all $x \in D$, where

$$\dot{h}(x, u) = L_f h(x) + L_g h(x) u, \quad u \in \mathcal{U}. \tag{6.14}$$

In particular,

$$L_f h(x) = \nabla h(x)^\top f(x) \qquad L_g h(x) = \nabla h(x)^\top g(x)$$

are the *Lie derivatives* of h with respect to the vector f and g, where $\nabla h(x)$ is the gradient of h with respect to x. Also, a continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and satisfies $\alpha(0) = 0$. While this function is said to belong to class $\mathcal{K}_\infty$ if it belongs to class $\mathcal{K}$, satisfies $a = \infty$, and $\lim\limits_{r \to \infty} \alpha(r) = \infty$.

If in Equation (6.13) the previously defined barrier function in Equation (6.11) is substituted, then the CBF condition becomes:

$$\inf_{u \in U} \left[L_f B(x) + L_g B(x) u \right] \leq \alpha \left(\frac{1}{B(x)} \right). \tag{6.15}$$

This function $B(x)$ is formally referred to as a *Reciprocal Barrier Function* because it satisfies

$$\inf_{x \in \text{Int}(C)} B(x) \geq 0, \qquad \lim_{x \to \partial C} B(x) = \infty.$$

This means that as the state nears the boundary ∂C, $B(x)$ becomes infinitely large, effectively stopping the system from leaving the safe set C. In particular, its definition is given as follows.

Definition 3. (*RCBF*) Consider the control system (6.8) and the set $C \subset \mathbb{R}^n$ defined by (6.9) for a continuously differentiable function h. A continuously a continuously differentiable function $B : \mathrm{Int}(C) \to \mathbb{R}$ called a *Reciprocal Control Barrier Function (RCBF)* if there exist class-$\mathcal{K}$ functions $\alpha_1, \alpha_2, \alpha_3$ such that for all $x \in \mathrm{Int}(C)$:

$$\alpha_1\big(h(x)\big) \leq B(x) \leq \frac{1}{\alpha_2\big(h(x)\big)}, \tag{6.16}$$

$$\inf_{u \in U} \Big[L_f B(x) + L_g B(x) u - \alpha_3\big(h(x)\big) \Big] \leq 0. \tag{6.17}$$

The RCBF B is said to be locally Lipschitz continuous if α_3 and $\frac{\partial B}{\partial x}$ are both locally Lipschitz continuous.

Finally, it is defined the set of all control values that render $\mathcal{C}$ safe as:

$$K_{\mathrm{cbf}}(x) = \{u \in U \mid L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq 0\}. \tag{6.18}$$

And similarly:

$$K_{\mathrm{rcbf}}(x) = \{\, u \in U : L_f B(x) + L_g B(x)\, u \; - \; \alpha_3\big(h(x)\big) \leq 0\}.$$

CBFs provide the strongest conditions for ensuring safety, as they are necessary and sufficient under reasonable assumptions on $\mathcal{C}$, see [21] for more details.

After established that Control Barrier Functions provide the necessary and sufficient conditions for safety, the next step is to synthesize controllers that enforce these conditions. The goal is to achieve this in a minimally invasive way, meaning that the modifications to a nominal controller should be as small as possible while still ensuring safety. This leads to optimization-based controller, that can be formulated as follows:

$$\begin{aligned} u(x) = \operatorname*{argmin}_{u \in \mathbb{R}^m} \; & \frac{1}{2} \left\| u - k(x) \right\|^2 \\ \text{s.t.} \quad & L_f h(x) + L_g h(x)\, u \; \geq \; -\alpha\big(h(x)\big), \end{aligned} \tag{6.19}$$

where u represents the optimized control input, while $k(x)$ denotes the nominal control input. Equation (6.19) will be called as CBF-QP.

6.2.2 CBF-QP

As mentioned before, when CBFs are integrated with a nominal controller through a Quadratic Programming framework, safety is guaranteed at all times. In this approach, the QP ensures that the control input satisfies the barrier condition, while maintaining safety, which means guaranteeing the objective specified by the nominal controller. Whenever the safety and performance objectives conflict, the safety constraint takes precedence, modifying the performance of the nominal control, as shown in Figure 6.1.

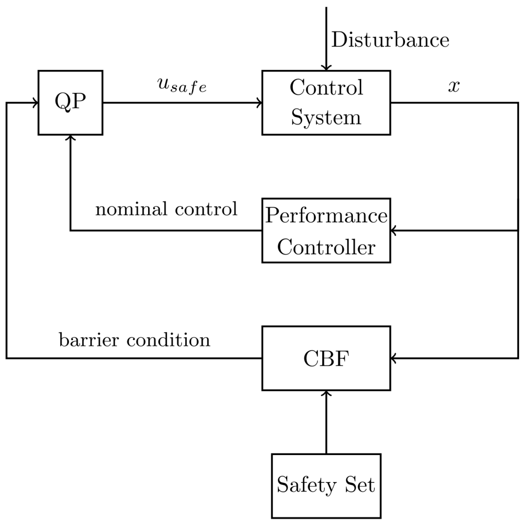

Figure 6.1: QP-based controller [27].

In particular, in Figure 6.1, the safety constraint and control performance objectives are unified through a barrier condition. Specifically, the controller u_{safe} is designed to satisfy the barrier condition, ensuring that the state of the control system x remains within the safety set, while remaining as close as possible to the given nominal control. As a result, the performance objective is met whenever it does not conflict with the safety constraint. The formal formulation of this will be provided in more detail below in the case under study.

Lane Keeping

Based on the studies presented in [21], it is now introduced the Lane Keeping (LK) problem. This is of particular importance in underwater scenarios since, during the path-following phase, an AUV may be subject to current disturbances. Although the surge and yaw dynamics can address these disturbances by combining their control and modifying the yaw reference, they do not respond as quickly as a direct control action on the sway. Since the AUV considered in this project can control its sway dynamics, this degree of freedom is taken into consideration to keep the lateral error, from the AUV position to the desired path, within acceptable bounds.

Specifically, the state to be regulated is the lateral position of the AUV (along the y-axis) relative to a desired reference line, which corresponds to the cross-track error, Equation (5.1.1), previously defined in the LOS guidance. Consequently, an additional control action is implemented to ensure that the cross-track error remains within prescribed bounds. This requirement can be reached by formulating the CBF-based QP defined in (6.19).

In this approach, the focus is on the lateral dynamics of the AUV, under the assumption that the vehicle maintains a constant longitudinal speed throughout the maneuver (Assumption 2, Section 3.2.2).

Problem Setup. Consider the two-state lateral model of the AUV:

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{Y_v}{m - Y_{\dot{v}}} \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m - Y_{\dot{v}}} \end{bmatrix} \tau_2. \tag{6.20}$$

Let the state vector be $\mathbf{x} := (y,\ \dot{y})$ where y denotes the lateral displacement in the fixed coordinate frame (NED), which is the cross-track error, and $\dot{y}$ is the lateral velocity. The control input τ_2 represents the lateral thrust force.

The objective of the Lane Keeping (LK) problem is to determine a lateral force input that keeps the AUV "centered" along the desired path. In particular, the system must satisfy specific hard control objectives while keeping the input constraints under consideration.

Hard Constraint. This constraint ensures that the lateral displacement of the AUV from the lane center remains below a specified value $y_{\max}$:

$$|y| \leq y_{\max}. \tag{6.21}$$

An arbitrary value of 0.2 meters was chosen for $y_{\max}$. Consequently, the hard constraint is $|y| \leq 0.2$. Equation 6.21 is referred as Lane Keeping - Hard Constraint (LK-HC). Motivated by the work done in [49], the following barrier function is considered:

$$h(x) = \Big(y_{\max} - \operatorname{sgn}(\dot{y})\, y\Big) \ - \ \frac{\dot{y}^2}{2\, a_{\max}}, \tag{6.22}$$

here $y_{\max}$ represents the maximum allowable lateral deviation, while the term $\operatorname{sgn}(\dot{y})y|$ specifies the direction of motion. The quadratic velocity term $\frac{\dot{y}^2}{2a_{\max}}$ ensures that the vehicle can always return to the safe region under the maximum allowable deceleration $a_{\max}$, both $y_{\max}$ and $a_{\max}$ are arbitrarily chosen. Then the feasible set is defined as $\mathcal{C} = \{x \in \mathbb{R}^n \mid h(x) \geq 0\}$ and for every $x \in \mathcal{C}$, the system can remain in $\mathcal{C}$.

It is considered a RCBF of the form:

$$B(x) = -\log\bigg(\frac{h(x)}{1 + h(x)}\bigg), \tag{6.23}$$

since this formulation ensure that the barrier function takes higher values near the boundary and lower values as the system moves away from it. The functions $h(x)$ and $B(x)$ are shown to satisfy the criteria of Equation (6.15), ensuring that the set $\mathcal{C}$ is controlled invariant. See [20] for more details.

Nominal Control. The nominal control strategy employed, given the linearized system dynamics in Equation (6.20), is the LQR. LQR is an optimal control technique that minimizes a quadratic cost function defined over the state and control input of the system. This method determines an optimal state-feedback gain by solving the algebraic Riccati equation, using the system dynamics.

The cost function is formulated as:

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt, \tag{6.24}$$

where the weight matrices $Q = Q^T \succeq 0$ and $R = R^T \succ 0$ define the relative importance of the state x and control input u, respectively.

The optimal control law is found by solving the algebraic Riccati equation. This equation is given by:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0. \tag{6.25}$$

The optimal state-feedback control law is then defined as:

$$u = -K_{LQR} x, \tag{6.26}$$

where the feedback gain matrix K_{LQR} is computed as $K_{LQR} = R^{-1}B^T P$. Here, P is a stabilizing solution to the Riccati equation, ensuring the optimal performance of the controller.

Here the first state, where the state vector is recalled to be $\mathbf{x} := (y,\ \dot{y})$, is given by the cross-track error of the LOS Guidance, which represents the lateral deviation of the AUV from the desired path. The LQR approach was chosen due to its efficacy in stabilizing dynamic systems at a desired equilibrium point. In this case, the objective is to regulate the cross-track error to zero, ensuring that the AUV remains as close as possible on the planned path.

Therefore, by unifying the safety hard constraint and the control soft constraint, the quadratic program (QP) is formulated as:

$$\begin{aligned} \mathbf{u}^*(x) = \underset{\mathbf{u}=[u,\delta]^T \in U_{lk}\times\mathbb{R}}{\arg\min} \quad & \frac{1}{2}\mathbf{u}^\top H_{lk}\,\mathbf{u} \\ \text{s.t.} \quad & A^{lk}_{fcbf}\,\mathbf{u} \ \le\ b^{lk}_{fcbf} \\ & u \ =\ -K_{LQR}x + \delta, \end{aligned} \tag{LK QP}$$

where $H_{lk} := \operatorname{diag}\{p_1, p_2\} \in \mathbb{R}^{2\times 2}$ is the weighting matrix where p_1 penalizes the control input $\mathbf{u}$ and p_2 penalizes the relaxation variable δ, which is introduced to ensure the feasibility of the QP. K_{LQR} is the optimal feedback gain defined in Equation

6.26 and finally the CBF constraints are defined by:

$$A^{lk}_{fcbf} = [L_g B(x),\, 0] \quad \text{and} \quad b^{lk}_{fcbf} = -\,L_f B(x) + \frac{\gamma}{B(x)}.$$

Note 1: Setting $\delta = 0$ makes the constraint imposed by the nominal controller “hard”. This means that the system is required to exactly perform as the nominal control without any relaxation. This means that, if no control input can be found that simultaneously satisfies both the LQR control objectives and the RCBF constraints, the QP becomes infeasible.

Note 2: It is important to observe that the formulation of the LK QP is identical to the one presented in Equation 6.19. Both QPs are designed with the objective of ensuring system stability and safety at the same time. They do this by keeping the control input as close as possible to the nominal control law and only switching to a safer control when necessary to avoid unsafe regions. This way, the system follows the usual behavior unless a safety issue arises.

Note 3: The function h has a relative degree of 1, where the relative degree is defined as the minimum order of the time derivative of the output y of a dynamical system in affine form, that is directly affected by the input u. Recalling that the sway dinamics is defined as:

$$\dot{v} = \frac{Y_v}{m - Y_{\dot{v}}} v + \frac{1}{m - Y_{\dot{v}}} \tau_2,$$

the h defined in (6.22) has relative degree of 1.If the function h has a relative degree greater than 1, then $L_g h = 0$ and the set $K_{\text{rcbf}}(x)$ becomes equals to U. When h has a relative degree $r \geq 2$, the use of High-Order Control Barrier Functions (HOCBFs) becomes necessary, see [50] for more details.

Chapter 7

Simulation Results

This chapter presents the results obtained from the work described in Chapters 4, 5 and 6. In particular, Section 7.1 will provide an overview of the simulation environment, focusing on the setup and configuration within Gazebo. Subsequently, Section 7.2 will present and analyze the results obtained from the simulations.

7.1 Simulation Environment

The simulation environments adopted throughout this project are Gazebo integrated with ROS and MATLAB/Simulink.

Gazebo is a robotics simulator that offers realistic environment and physics interactions between different systems in the simulation, allowing a high-fidelity testing in complex environments. For example, in a marine world, Gazebo can simulate currents and buoyancy, thereby replicating real-world conditions. On the other hand, ROS, which stands for Robot Operating System, though it is more a framework than an actual operating system for writing robot software, offers a collection of tools and libraries that help build and simulate various robotic applications. ROS facilitates communication between different software components (or nodes), supports hardware abstraction, and offers a large and active open-source community.

In this study, the simulation related to path planning, guidance, and control has been implemented using Gazebo, leveraging its capability of simulating dynamic environment and offering a visual realistic scenarios for testing. In contrast, the part of the project dealing with safety-critical control, specifically the implementation of CBFs, was realized in MATLAB and Simulink, which is ideal to simulate functionality in control systems, and thus to analyze safety-critical aspects. The use of these two types of tools makes it possible to harness the benefits of both simulators. That means, simulating through Gazebo/ROS provides an interactive and real-time world of the AUV behavior underwater, while MATLAB/Simulink offers a better understanding of control system analysis and validation.

7.1.1 Gazebo

Figure 7.1 provides an overview of the ROS architecture implemented throughout the project.

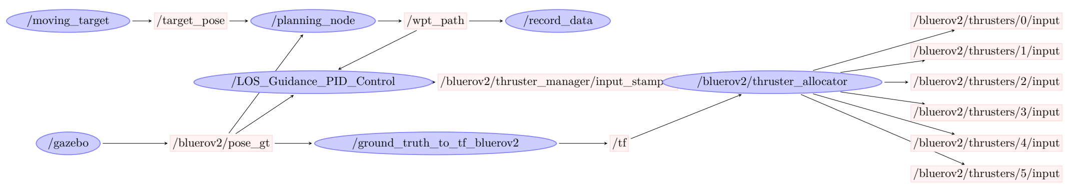
/moving_target
/target_pose
/planning_node
/wpt_path
/record_data
/bluerov2/thrusters/0/input
/bluerov2/thrusters/1/input
/LOS_Guidance_PID_Control
/bluerov2/thruster_manager/input_stamped
/bluerov2/thruster_allocator
/bluerov2/thrusters/2/input
/bluerov2/thrusters/3/input
/gazebo
/bluerov2/pose_gt
/ground_truth_to_tf_bluerov2
/tf
/bluerov2/thrusters/4/input
/bluerov2/thrusters/5/input

Figure 7.1: Architecture overview of the ROS nodes and topics. Blue indicates nodes, pink indicates topics.

In ROS, *nodes* are individual processes that perform specific tasks, here for example the planning and control. *Topics* are communication channels that allow nodes to exchange data using a publisher-subscriber model.

In particular:

- Informations on the moving target: The `/moving_target` node publishes each 6 seconds the pose of the target on the `/target_pose` topic.
- Path Planning RRT* Execution: The `/planning_node` subscribes to `/target_pose` and computes an optimal path to the target, publishing the generated waypoints on `/wpt_path`.
- LOS Guidance and PID Control Execution: The `/LOS_Guidance_PID_Control` node processes the planned path and calculates control commands to move the vehicle.
- Data Logging: The `/record_data` node captures and logs system information.
- Coordinate Transformations and Thrust Allocation: The `/ground_truth_pub_to_bluerov2` node provides real-world data to the `/bluerov2_thruster_allocator` after processing transformations via the `/tf` package. The `/bluerov2_thruster_allocator` distributes the computed thrust commands to each individual thruster.

The Gazebo implementation of the BlueROV2 was carried out using the open-source *UUV Simulator*, which is an extension of the Gazebo simulator designed specifically for underwater scenarios. This approach enables realistic hydrodynamic modeling and sensor simulation (such as sonar and Inertial Measurement Unit, IMU). In Gazebo, a robot is represented by a model defined through SDF/URDF files that include its physical properties (such as mass, inertia, and geometry), sensor configurations and actuator specifications. In addition, plugins are used to simulate dynamics and environmental interactions. More details on how the BlueROV2 was implemented in the *UUV Simulator* can be found in [51]. The BlueROV2 is spawned in Gazebo as shown in Figure 7.2.

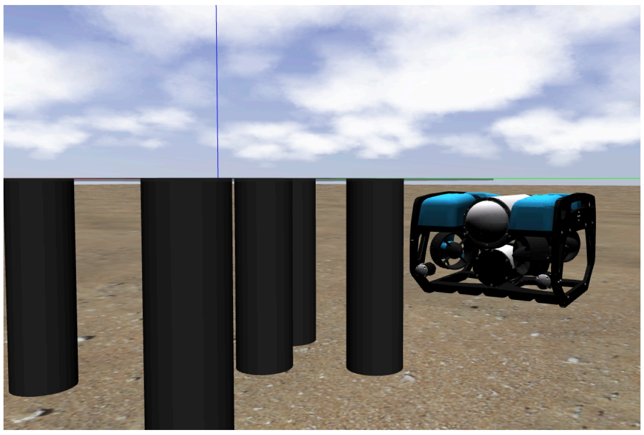

Figure 7.2: Underwater windfarm simulation with BlueROV2 deployed in Gazebo.

In Gazebo, the underwater windfarm was simulated with 7 wind turbines within a 20×20 m^2 area. Note that these values, as well as the arrangement of the turbines, were chosen arbitrarily. The layout of the turbines and the environment from a top view is shown in Figure 7.3.

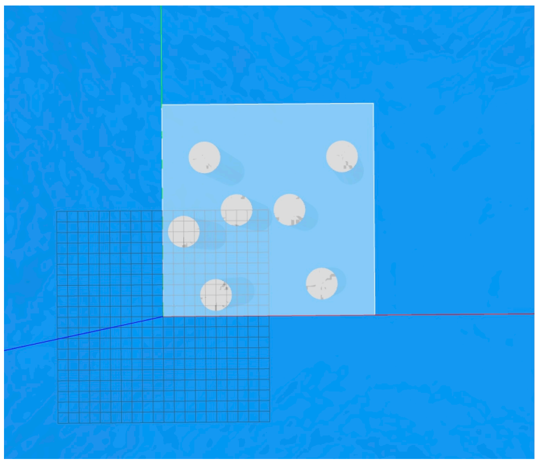

Figure 7.3: Top view of the simulation environment.

The information regarding the display of the reference trajectory and the representation of the moving intruder is shown in RViz. RViz (ROS Visualization) is a visualization tool used to display sensor data and robot states. RViz can be seen in Figure 7.4.

In Figure 7.4, it can be observed that the world, NED (the AUV) reference frames and the the static obstacles (the blue circles), are represented. In the reference frames, the red line represents the x-axis, the green line represents the y-axis and the blue line is the z-axis, which is directed out of the plane. In addition, the moving target is depicted with the red arrow. Subsequently, the results obtained in this environment are presented.

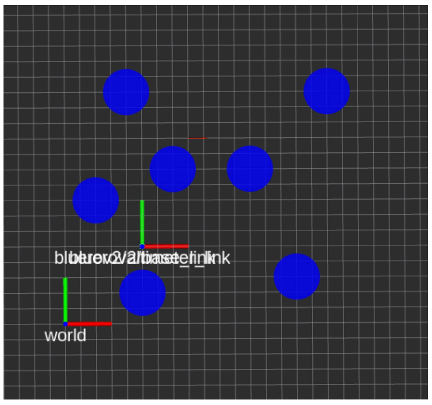

Figure 7.4: RViz simulation environment.

7.2 Simulation Outcome

As previously mentioned, the simulation was carried out partially in ROS/Gazebo and partially in MATLAB/Simulink. Sections 7.2.1 and 7.2.2 present the results obtained in ROS/Gazebo, where the RRT* is first showed, then the LOS guidance, and PID control for surge and yaw are simulated. Section 7.2.3 presents the simulation of the CBF in MATLAB/Simulink.

7.2.1 Path Generation

The RRT* path search is illustrated in the Figure 7.5. The configuration parameters are provided in Table 4.1. The algorithm explores a 20×20 m² configuration space, building the tree shown in green. The optimal path found is depicted in red. The start point (Start) is indicated in blue, while the goal point (Goal) is shown in green.

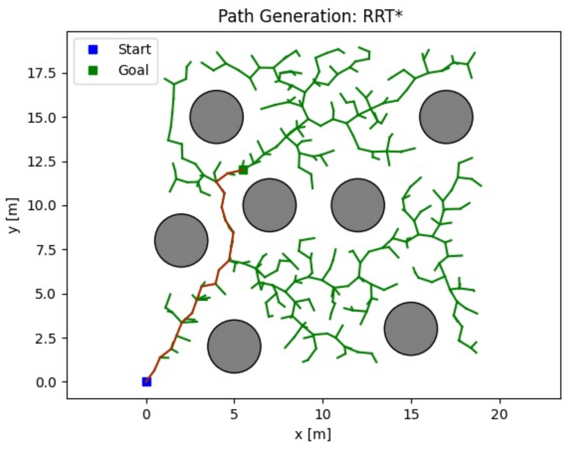

Figure 7.5: RRT* Path Search.

Additional simulations were performed starting from different initial positions as shown in Figure 7.6. In all cases, the path generated respects the minimum turning radius constraint of the AUV. Which has been chosen arbitrarily to be 90°.

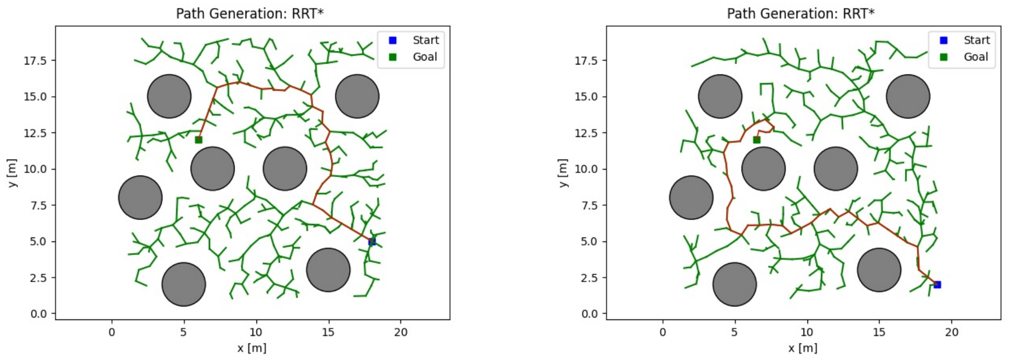

Figure 7.6: RRT* Path Searches.

7.2.2 Path Following

Before simulating the path following of the trajectory generated with a moving target, a *zigzag* trajectory was tested to validate the LOS and PID parameters. The *zigzag* trajectory was used to assess the maneuver performance of the yaw model. The trajectory was generated using 12 points, with each step of 1 meter. The LOS parameters, in Table 5.1, and the PID gains chosen, Section 6.1.1, appears to be satisfactory.

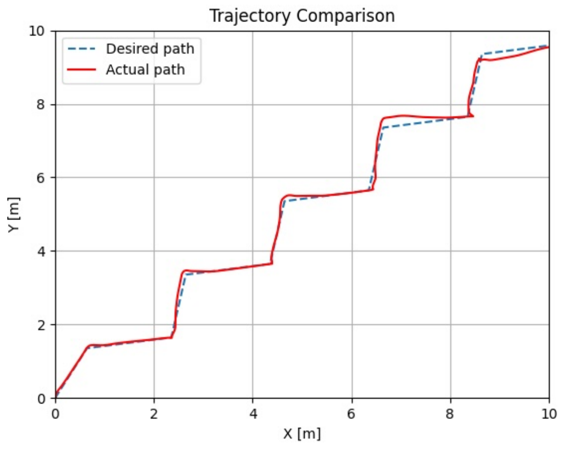

Figure 7.7: Trajectory comparison over time.

Figure 7.7 shows the comparison between the desired and the actual trajectory over time. The starting point is $(0, 0)$ with an initial yaw angle of 1.57 rad (90 deg), as illustrated in Figure 7.8, which depicts the yaw error over time. The desired speed was set to 0.2 m/s, that was done due in order to limit the motion coupling. Additionally, Figure 7.9 presents the cross-track error. This parameter is the one that it is taken into account for the evaluation of the CBF, which means for the decision of the maximum allowable lateral distance. This error oscillates around 0.2 m, a value that is considered to be reasonable.

Once the LOS and PID parameters were evaluated, the following simulation demon-

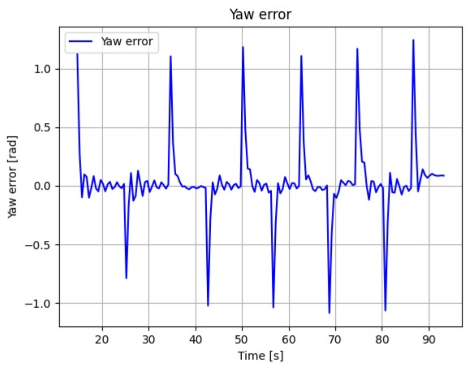

Figure 7.8: Yaw error.

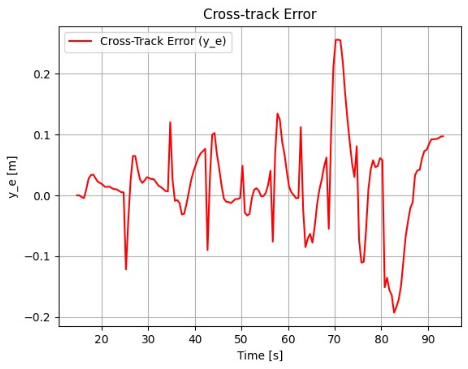

Figure 7.9: Cross-Track error.

strates the path planning generation using the RRT* algorithm with the values specified in Table 4.1, along with the path following using the LOS and PID parameters provided in Table 5.1. The simulation was conducted by implementing a moving target whose position was updated every 6 seconds, with the target moving at a constant speed of 0.1 m/s computing a straight line path starting from position (6.0, 12.0).

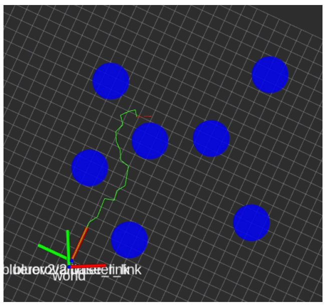

Figure 7.10: RViz Path 1.

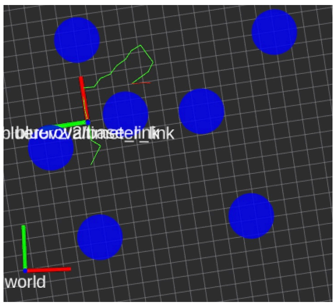

Figure 7.11: RViz Path 3.

Figures 7.10 and 7.11 show that the path (the green line) generated by the RRT* is computed based on the current positions of both the AUV and the target. The time needed to compute a new path, from the moment the AUV detects the updated target position to the generation of the next trajectory, is approximately 2 seconds.

Note: The approach direction towards the target (i.e., whether the AUV arrives from the front, back, or side) has not been considered in the path generation. As a result, the AUV reaches the target with an arbitrary orientation.

Figure 7.12 shows a comparison between the actual path executed by the AUV versus the desired paths generated during the overall simulation. The actual path of the AUV is highlighted in black, while the dashed lines indicate the various paths generated by the RRT*, based on the AUV's position at each moment. The starting position of the AUV is in (0,0), represented by a black cross, while the red crosses represents the position of the target moving throuout the simulation. The target is moving in a straight line starting from position (6.0,12.0) and moving towards positive values of x until the AUV reaches it.

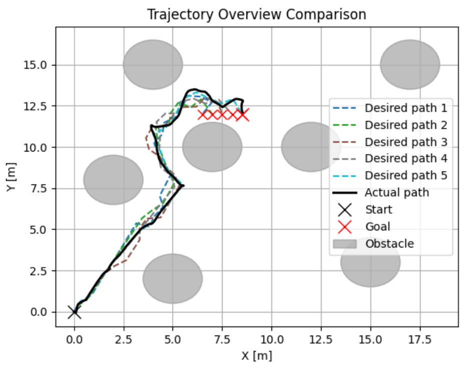

Figure 7.12: Simulation 1: Trajectory comparison in time with all the generated paths.

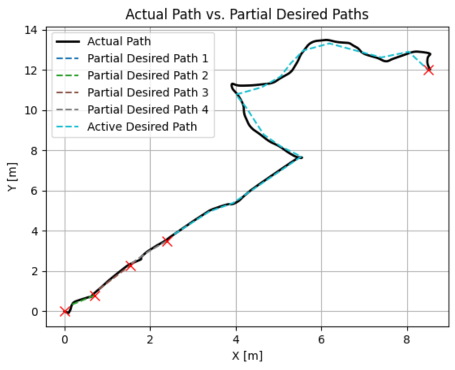

Figure 7.13: Simulation 1: Trajectory comparison in time with the partial desired paths.

In particular, Figure 7.13 shows the partial desired path that the AUV follows until the target updates its position. At that point, a new desired path is computed using RRT*, and the AUV switches from the old path to the newly generated one. This process is highlighted by the red crosses in the figure, indicating the points where the desired path is updated.

Figures 7.14 and 7.15 provide a zoomed view of the trajectory shown in Figure 7.19. It can be observed that the trajectory is smoother in the first section, where it follows an almost straight path, while it becomes less accurate during a turn in the second part of the trajectory.

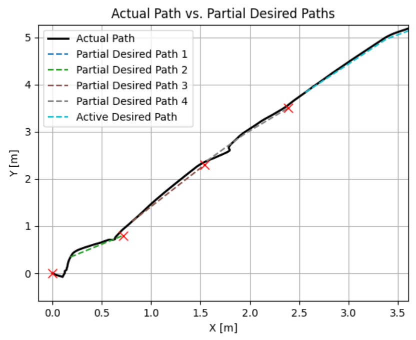

Figure 7.14: Zoomed view of the first part of the tracked trajectory.

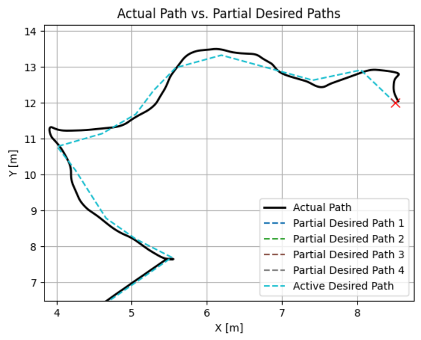

Figure 7.15: Zoomed view of the second part of the tracked trajectory.

Finally, Figures 7.16 and 7.17 illustrate the yaw error and the cross-track error, respectively. The yaw error remains generally stable, with spikes corresponding to the adjustments in yaw angle required to follow the path. It can be observed that the error is more evident in the second part of the trajectory, where the AUV follows an almost circular path. This increased oscillation is due to the continuous adjustments required to follow the curved trajectory, making the control corrections more frequent and pronounced.

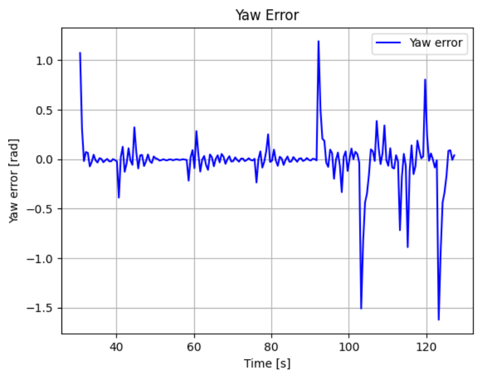

Figure 7.16: Simulation 1: Yaw Error.

Meanwhile, the cross-track error oscillates around ± 0.2 meters in the first part of the trajectory. However, in the second part, during the turning phase, this error also increases significantly. This behavior highlights the fact that during the execution of a non-straight trajectory, it is necessary to introduce a lateral bound to ensure a good trajectory following.

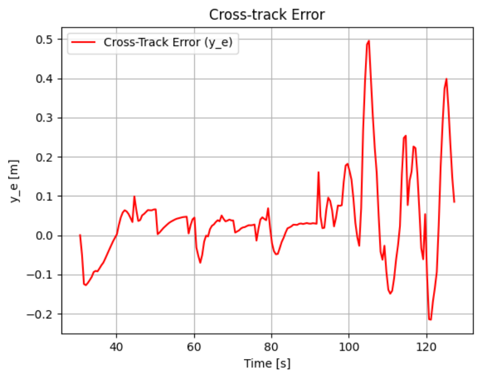

Figure 7.17: Simulation 1: Cross-track Error.

Note: The initial time displayed in Figures 7.16 and 7.17 appears to start at around 30 seconds. This happens because the control algorithm was not executed at the same time as the Gazebo simulation and the path planning process.

The sequence of operations is as follows:

1) *Launch of Gazebo environment*: The simulation environment is launched, which includes spawning the AUV and the underwater wind farm. This marks the start of the simulation timer.

2) *Launch of the path planning algorithm*: The path planning algorithm is then activated to compute a feasible trajectory for the AUV.

3) *Launch of the control (path following) algorithm*: When a valid trajectory has been generated the control algorithm is executed and the AUV starts following

the path.

Since the control algorithm is started only after the trajectory has been computed, the graphs in Figures 7.16 and 7.17 do not record the first 30 seconds of simulation time. Consequently, the total duration of the simulation is approximately 90 seconds.

Note: The PID controller in Gazebo is implemented in discrete time as in real world. The integration follows the *Forward Euler method*, approximating the integral as a sum of discrete areas with $\Delta t = 0.2$. The derivative is estimated using a backward difference, introducing discretization effects.

A second simulation was conducted, this time initializing the AUV from a different starting position.

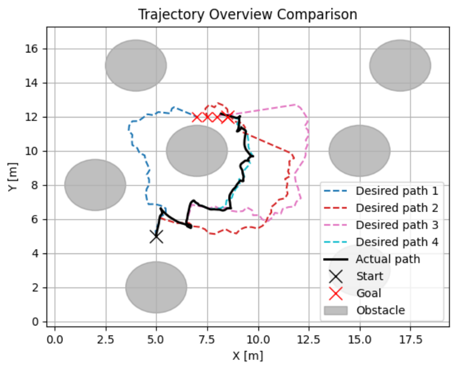

Figure 7.18: Simulation 2: Trajectory comparison in time with all the generated paths.

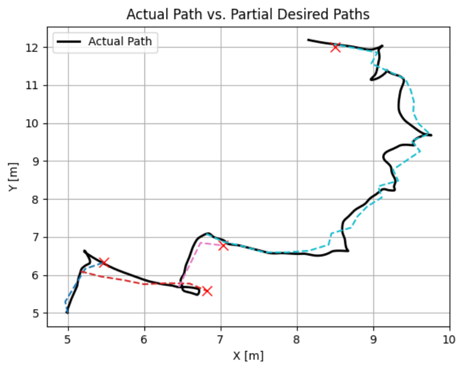

Figure 7.19: Simulation 2: Trajectory comparison in time with the partial desired paths.

It can be observed that, due to this initial placement, the first generated path,*Path 1*, differs from the subsequent ones. Specifically, in *Path 1*, the RRT* determined that the shortest route to the target was to the left of the obstacle. However, from *Path 2* onward, the updated position of the AUV caused the planner to generate a trajectory where the shortest path to the target was now to the right of the obstacle. This change of path generation highlights how the position of the AUV influences the generated path.

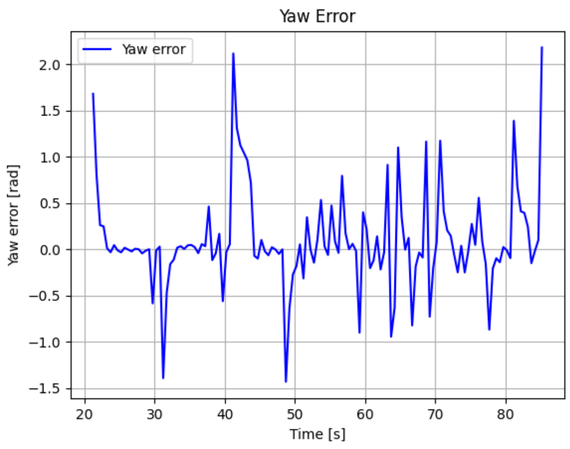

Figure 7.20: Simulation 2: Yaw Error.

Once again, it can be observed that during the path following phase of a non linear trajectory, both the yaw error and the cross-track error show greater oscillations compared to those observed in a straight-line path. A possible approach to mitigate the cross-track error is to implement a control strategy based on Control Barrier Functions, as previously introduced. The implementation and the analysis of those is shown in the following section.

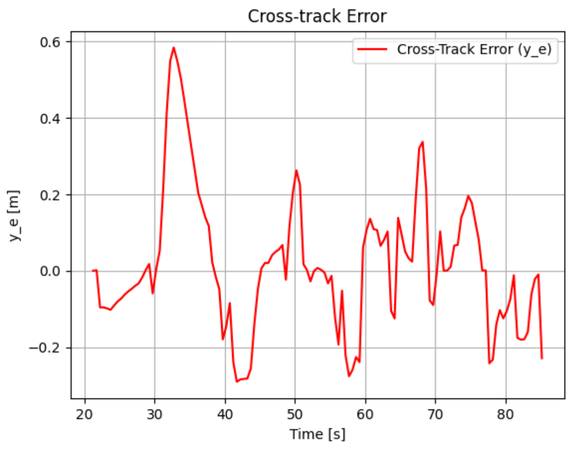

Figure 7.21: Simulation 2: Cross-track Error.

7.2.3 Safety Constraints

The study on the barrier function was conducted in MATLAB/Simulink. In particular, the sway dynamics of the AUV is analyzed and a control based on CBF-QP is implemented, as described in Section 6.2.2.

For the nominal control design, the LQR gain matrix K was computed using the weighting matrices $Q = \text{diag}(100, 10)$ and $R = 1$, while for the QP parameters, the matrix H was chosen as $H = \text{diag}(1\text{e}{-5}, 10)$ and $F = [0 \quad 0]$.

In order to deal with the Lane Keeping formulation problem, it is recalled that the Reciprocal Control Barrier Function adopted is the one defined in Equation 6.23 where $h(x)$ is defined in Equation 6.22. The selected barrier function is designed to ensure safety. To achieve this, $B(x)$ must remain positive within the safe set, where $h(x) > 0$. As the system state approaches the boundary of the safe set, meaning $h(x) \to 0$, the function $B(x)$ tends to infinity, preventing the system from violating safety conditions. This behavior imposes a strong penalty while the system is near the bound limit, imposing the system not to leave the safe region.

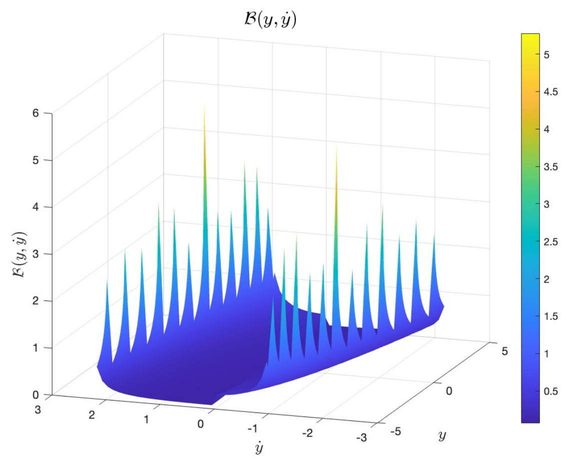

Figure 7.22: Behaviour of the Reciprocal Control Barrier Function.

The RCBF behaviour can be analyzed in Figure 7.22, where the function assumes higher values near the lane boundaries, ensuring that the control action maintains the vehicle within the safe set.

In particular, the barrier parameters have been selected as: $y_{\max} = 0.2$ and $a_{\max} = 7.8$. Here $a_{\max}$ is the maximum acceleration allowed by the AUV, while $y_{\max}$ is the maximum lateral displacement that the AUV must maintain during the path-following phase. The QP was implemented using the `quadprog` command in MATLAB.

Assumptions and Considerations.

- The AUV is assumed to follow a straight-line path, meaning the angular velocity $r = 0$, while maintaining a constant surge speed U. It is assumed a speed of 0.2 m/s.
- It is assumed that the AUV has the shape of a rectangular prism.
- The external disturbance is modeled as a force acting perpendicular to the AUV. This force represents the effect of the ocean current. It is assumed to behave like a drag force F_D acting only in the lateral direction and perpendicular to the lateral surface of the AUV. It is modeled using a quadratic drag

equation, as cited by [52] and [53], with the following form:

$$F_D = \frac{1}{2}\rho v^2 C_D A,$$

where ρ [kg/m^3] is the density of water, v [m/s] is the relative velocity between the AUV and the current, C_D is the drag coefficient, and A [m^2] is the reference area exposed to the flow. In particular,

- $\rho = 1000$ kg/m^3 has been used for the density of water.
- Assuming no sway velocity of the AUV, the velocity component v corresponds to the velocity of the current itself. Different values have been assigned to this parameter, considering a range from 0.2 to 0.7 m/s. This range is considered reasonable compared to the dimension and thrust power of a system like the BlueROV2.
- $C_D = 1$. This drag coefficient depends on the shape of the body and the Reynolds number. For a system with a cube form this could be approximated to 1.
- $A = 0.34 \times 0.25 = 0.085$ m^2, again considering the AUV as a full cube.

• One aspect to consider during the simulation is the control input limit. This is expressed in Newtons [N] and represents the control action needed exclusively to the sway dynamics. This input must be constrained by the maximum power that each thruster can provide. The thrusters used in this project are the T200 from *BlueRobotics* and according to the datasheet, these thrusters can generate a maximum thrust of approximately 40 N (see Appendix A). Considering the thruster configuration in the BlueROV2, previously shown in Figure 3.3, it can be seen that the four thrusters responsible for horizontal motion are inclined at 45 degrees. Therefore, each thruster can provide a maximum force of approximately 28 N in the sway direction. Therefore, the total available force for sway dynamics is 112 N.

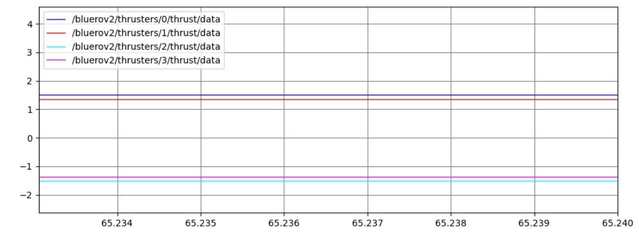

Figure 7.23: Thurster input request in Gazebo.

However, since the AUV is moving forward, it is important to determine the actual force available for sway motion. An analysis of the thrust input required by the thrusters was conducted during a Gazebo simulation, where the AUV was computing a straight trajectory starting at a velocity of 0.0 and reaching a desired velocity 0.2 m/s.

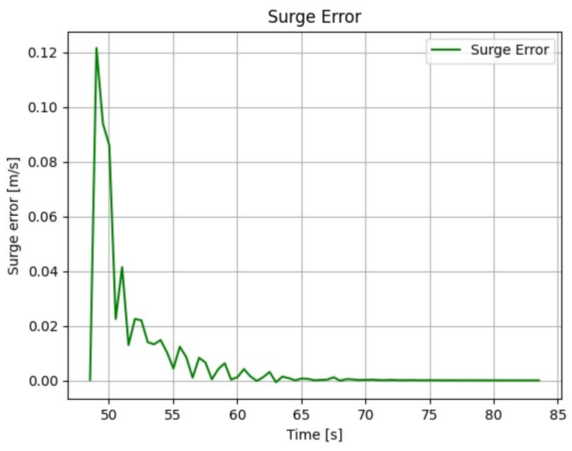

Figure 7.24: Surge error in time in Gazebo.

As shown in Figure 7.23, the four thrusters operate in pairs. Under steady-state conditions (see Figure 7.24), each thruster provides a thrust of 2 N. More specifically, thrusters 0 and 1 (front-right and rear-left) generate positive thrust values, while thrusters 2 and 3 (front-left and rear-right) generate negative thrust values. This configuration is done to ensure the AUV to compute a yaw moment. Thrusters 5 and 6 were not displayed as they are responsible for vertical motion, therefore are equale to 0 and not considered throughout this project. This simulation was conducted using the PID controller before analyzed.

Therefore, the available input force for the sway operation can be set to be approximately 100 N. Recalling that, after the calculation of this control input, the Thruster Allocation Matrix distributes this force among the individual thrusters.

Simulations.

To evaluate the behavior of the sway dynamics under the influence of current during the path following phase, the following simulations have been performed:

1) Initial conditions: $y_0 = 0.1$, $\dot{y}_0 = 0.2$

   Disturbance: 1) A current disturbance with a velocity of $v_c = 0.5$ m/s is applied starting at time $t = 10$ seconds. 2) An additional current disturbance with a velocity of $v_c = 0.3$ m/s is applied at a opposite direction (which means decreasing the disturbances) starting at time $t = 15$ seconds lasting for 5 seconds.

2) Initial conditions: $y_0 = 0$, $\dot{y}_0 = 0$

   Disturbance: 1) A current disturbance with a velocity of $v_c = 0.5$ m/s is applied starting at time $t = 0$ seconds. 2) An additional current disturbance with a velocity of $v_c = 0.3$ m/s is applied in the same direction (which means increasing the disturbances) starting at time $t = 5$ seconds lasting for 15 seconds.

3) Initial conditions: $y_0 = 0$, $\dot{y}_0 = 0$

   Disturbance: 1) A current disturbance with a velocity of $v_c = 0.2$ m/s is applied starting at time $t = 2$ seconds. 2) An additional current disturbance with a velocity of $v_c = 0.1$ m/s is applied at time $t = 10$ seconds lasting for 2 seconds. 3) An additional current disturbance with a velocity of $v_c = 0.3$ m/s is applied on the opposite direction at time $t = 15$ seconds lasting for 5 seconds.

Note: The initial conditions has been chosen such that the system starts in a safe state. Indeed, if the system were set to start in an unsafe state, the QP would not get a feasible solution, thus it would not guarantee that the system remains within the prescribed limits. This implies that CBFs are not responsible for driving the system into the safe set. Instead, given that the system starts within the safe set, they ensure that it remains inside this region.

In order to visualize the safe set, Figure 7.25 shows the region defined by the the red dashed and the blue curves where it is guarantee that if $y = y_{\max}$, then $\dot{y} < 0$, and if $y = -y_{\max}$, then $\dot{y} > 0$.

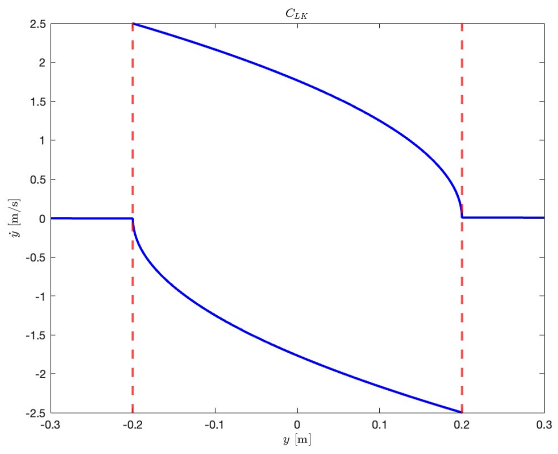

Figure 7.25: Projection of the safe set onto the $(y, \dot{y})$ plane.

Note: In the equation of the barrier function that has been adopted, the presence of the *sign* function can cause some problems during the implementation. In fact, the discontinuity of the sign function can cause numerical issues. The sudden transition at zero causes sudden changes in the control input.

To address this issue, the *sign* function has been substituted with an approximation, which is the *sigmoid* function. This alternative formulation should ensure a continuous transition between values. The *sigmoid* function is parameterized by a gain factor k, which corresponds to the level of approximation to the *sign* function. By imposing a sufficiently high values of k and by scaling the function within the range $[-1, 1]$, the *sigmoid* function was used in the following simulations.

1) In the first simulation, the initial conditions are set to $y_0 = 0.1\,\text{m}$ and $\dot{y}_0 = 0.2\,\text{m/s}$. Two different disturbances are applied in sequence, the first is applied at time $t = 10$ with a velocity of the current of $v_c = 0.5$ m/s, after 5 seconds a second disturbance is applied, on the opposite direction, which means a decreasing value of the disturbance on the BlueROV2 with a velocity of $v_c = 0.3$ m/s, lasting 5 seconds. After this disturbances, it is supposed to return in an ideal situation where no currents are present.

In Figure 7.26 it is presented a comparison between the lateral displacement of the AUV with and without the application of the CBFs. The blue dashed line represents the lateral position of the AUV under the control of the only LQR control law, without the additional constraint provided by the CBFs. The red line shows the lateral position when the CBF constraint is applied. It can be seen that without the CBF, the AUV exceeds the lateral limit of 0.2 m. However, when the CBF constraint is incorporated, the maximum lateral displacement is maintained, ensuring safe operation. It can be notices that the period of time between $t = 10$ and $t = 20$, which is the period od time of the presence of disturbance, the lateral displacement is kept to the maximim limit. Whenever the disturbance is no longer present (i.e., the vehicle is no longer influenced by the currents), the lateral position is brought back to 0.

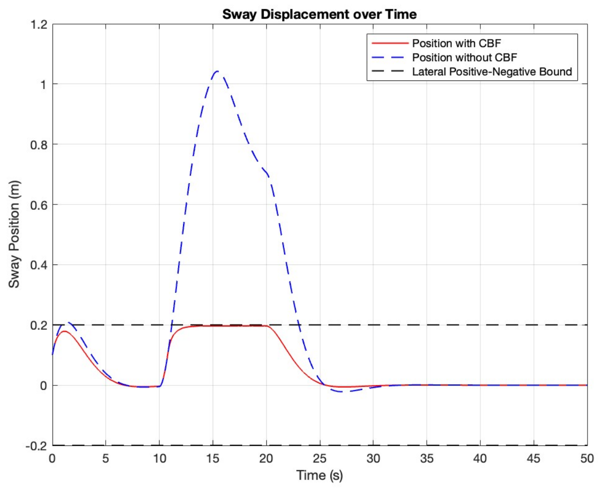

Figure 7.26: Simulation 1: Comparison of sway position obtained with and without CBF.

Figure 7.27 presents a comparison of the lateral velocities of the AUV with and without the use of CBFs, similar to the analysis performed for the position. The blue dashed line represents the lateral velocity when only the LQR controller is applied, while the red line corresponds to the velocity when the CBF constraint is applied. In the figure, a chattering effect appears at around $t = 13$ seconds. Despite replacing the *sign* function with a *sigmoid* function, the chattering persists. This happens because the chosen scaling factor k for the sigmoid function has been set to a very high value to approximate as close as possible the behavior of the *sign* function. Since the AUV operates within a small range, given that the bound limit is set to 0.2, reducing k too much would fail to replicate the behavior of the *sign* function.

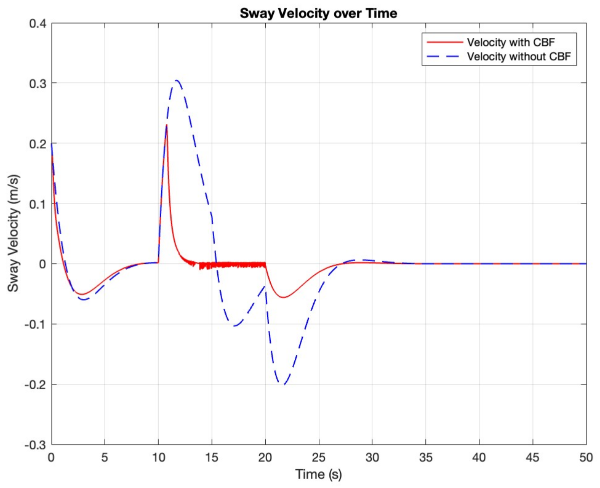

Figure 7.27: Simulation 1: Comparison of sway velocity obtained with and without CBF.

Indeed, Figure 7.28 illustrates the evolution of the barrier function over time. As observed in the velocity profile, a chattering effect is present at $t = 13$ seconds until the disturbance goes back to zero. Additionally, due to the initial position of the vehicle, which is close to the boundary, the barrier function momentarily activates at the beginning before stabilizing at a constant value when the position returns to zero. Then, once the disturbance is present, the barrier function is reactivated.

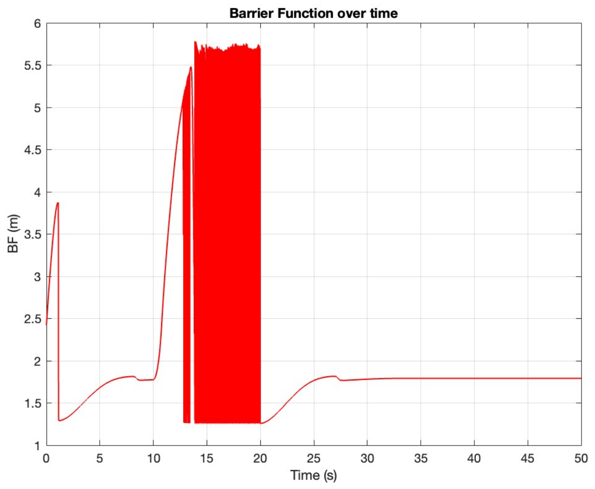

Figure 7.28: Simulation 1: Barrier Function behavior.

Furthermore, the figure provides a representation of the behavior of the Reciprocal

CBF. Initially, when the starting position of the AUV is nonzero but still within the boundary, the barrier function assumes a moderate value, with a value of approximately 3.8. As the AUV approaches the boundary, the CBF rapidly increases, with a maximum at 5.6. This shows the nature of the barrier, which assumes incremental higher values when the system is gettiing closer to the boundaries and approaches zero when it is far away. In fact, when the disturbance returns to zero and the position returns to zero, the barrier function stabilizes at a constant value, which menas that the nominal control action is not modified. However, the presence of chattering implies that further refinements in the *sigmoid* function might be required.

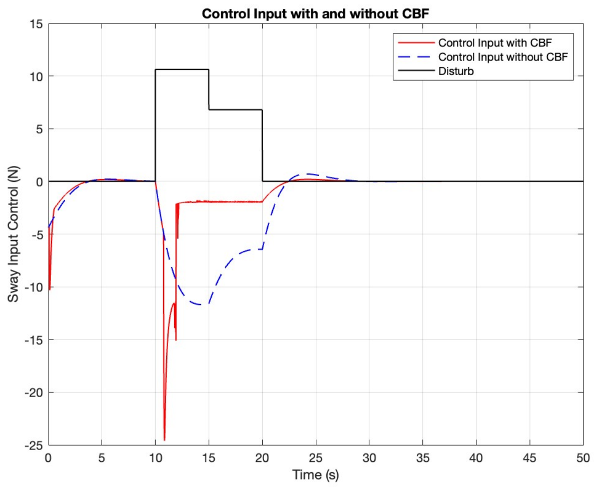

Figure 7.29: Simulation 1: Comparison of the control input obtained with and without CBF.

Figure 7.29 presents a comparison between the control input generated only by the LQR controller and the one optimized using the Barrier Function. The dashed black rectangular line represents the disturbance applied to the system, the dashed blue line corresponds to the CBF-based QP input and the red line represents the control input from the LQR controller.

Initially, the two control inputs have a similar, but not identical, behavior. This difference comes up because the AUV starts from a position that is pretty close to the boundary. As already explained, the constraint is imposed in order to guarantee the system to stay in the safe set. When the disturbance is introduced, there is a significant difference between the two control inputs. In particular, the control input generated by the constrained reaches much larger value compared to the LQR input.

Since the barrier function enforces the AUV to be in the 0.2 bound, the optimized control input keeps a constant value to ensure the position to be within this boundary, while, the LQR, which objective is to to bring the state y back to zero without constraints, produces control values that change depending on the disturbance intensity. On the other hand, the CBF control input remains constant regardless the change on the disturbance. This is because, once the system reaches a limiting state, the control input must keep the vehicle to stay in the limit.

As soon as the disturbance goes back to zero, both the control inputs converge back to zero, which means that the system has returned to the desired position of $y = 0$, meaning a null cross-track error.

It is important to also analyze the maximum value that the CBF control has reached. An absolute value of 25 N is considered to be completely within the maximum range of 100 N, therefore it is considered an acceptable value.

2) In the second simulation, the initial conditions are set to $y_0 = 0\,\mathrm{m}$ and $\dot{y}_0 = 0\,\mathrm{m/s}$. Two different disturbances are applied in sequence, the first is applied at time $t = 0$ with a velocity of the current of $v_c = 0.5$ m/s, after 5 seconds a second disturbance is applied, on the same direction, which means an increasing value of the disturbance on the BlueROV2 with a velocity of $v_c = 0.3$ m/s, lasting 15 seconds. After this disturbances, it is supposed to return in an ideal situation where no currents are present.

Again, Figure 7.30 shows the position of the AUV, with and without the constraint on the CBF. It can be observed that, given an initial position of 0 and a disturbance entering at time $t = 0$, the barrier (shown in Figure 7.32) is activated at $t = 0$. However, until approximately $t = 3$, the barrier gradually increases its value since the AUV remains within the boundary but has not yet reached its limit. Once the AUV reaches the boundary, the barrier reaches its maximum values, before stabilizing again when the disturbance goes back to zero.

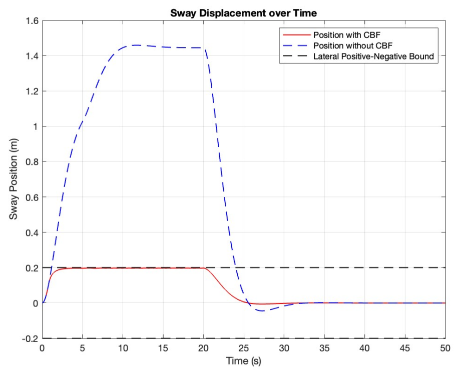

Figure 7.30: Simulation 2: Comparison of sway position obtained with and without CBF.

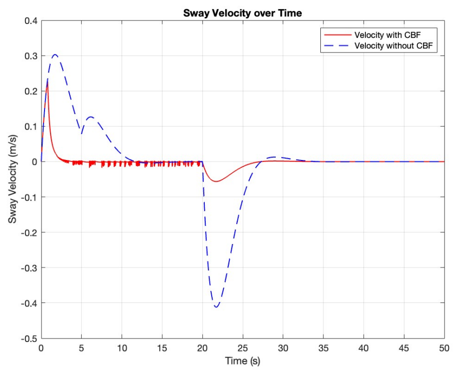

Figure 7.31: Simulation 2: Comparison of sway velocity obtained with and without CBF.

The effect of the barrier is particularly evident in Figure 7.31, which shows the velocity. Here, chattering is present during the same time interval in which the barrier is active. Furthermore, it is once again highlighted that the position is successfully kept within the limits throughout the entire duration of the disturbance.

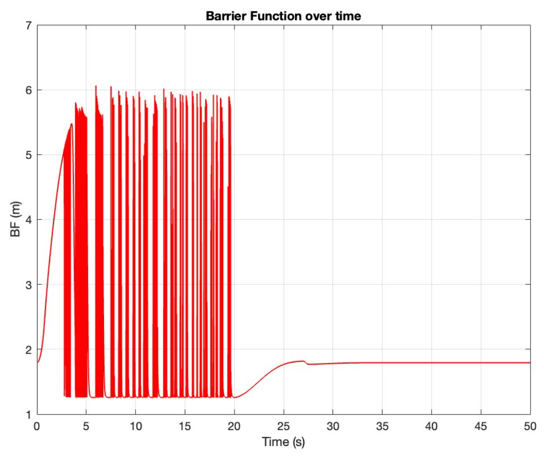

Figure 7.32: Simulation 2: Barrier Function behavior.

Figure 7.33 once again presents a comparison between the control input u generated with and without the CBFs. It can be observed that, since the disturbance enters at time $t = 0$, the control input are very different from the beginning. As in the previous case, the optimized control input reaches high values, and switching behavior is again evident around $t = 3$.

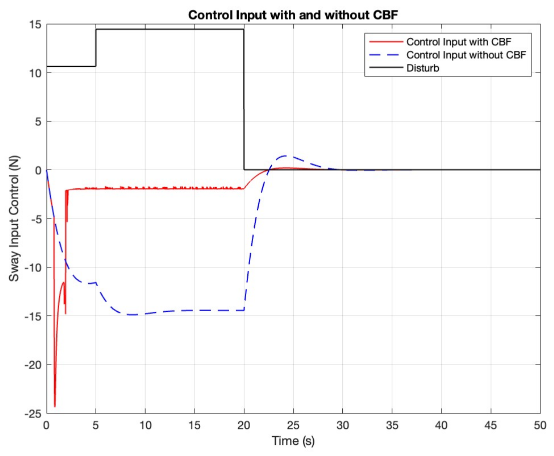

Figure 7.33: Simulation 2: Comparison of the control input obtained with and without CBF.

The value of 25 N is once again considered within the acceptable range to ensure that the AUV maintains the desired velocity and is able to perform the lateral correction, thereby ensuring that the system remains within the desired bounds.

3) In the last simulation, the initial conditions are set to $y_0 = 0\,\text{m}$ and $\dot{y}_0 = 0\,\text{m/s}$. Three different disturbances are applied in sequence, the first is applied at time $t = 2$ s with a velocity of the current of $v_c = 0.2$ m/s, after 8 seconds a second disturbance is applied, on the same direction, which means an increasing value of the disturbance on the BlueROV2, with a velocity of $v_c = 0.1$ m/s lasting for 3 seconds. A third current disturbance with a velocity of $v_c = 0.3$ m/s is applied on the opposite direction at time $t = 15$ seconds lasting for 5 seconds.

Figures 7.34 and 7.35 show the position and velocity of the AUV, respectively. It can be observed that the position is maintained with the activation of the LQR controller alone during the first two disturbances. However, for the third disturbance, which involves a current with a velocity of 0.3 m/s, the inclusion of CBF becomes necessary. Also, it is important to highlight that the two position displacement are identical in the initial phase, as the position stays within the safe bound. Similarly, the velocity follows the same profile in the first phase but becomes differen whenever a higher disturbance comes.

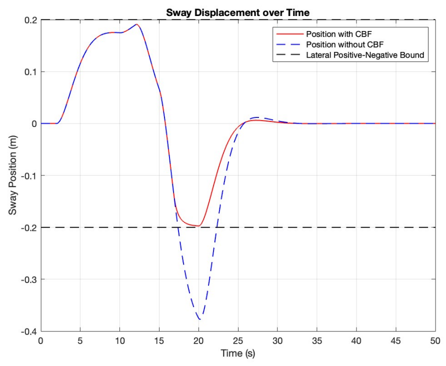

Figure 7.34: Simulation 3: Comparison of sway position obtained with and without CBF.

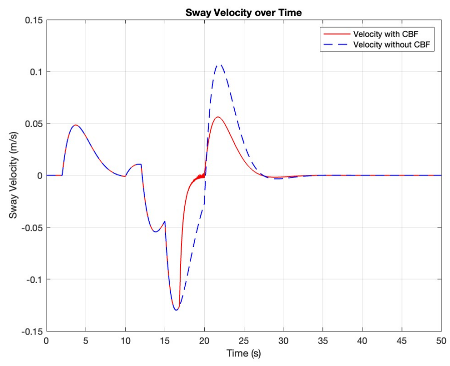

Figure 7.35: Simulation 3: Comparison of sway velocity obtained with and without CBF.

Figures 7.36 show the comparison between the two different inputs. It can be observed that, as previously stated, with the two disturbances, since the AUV is within the safe bound, the two control inputs are equal, since it is not required a higher control input to stay within the limits. Note that this implies that the optimization delta of the QP is equal to zero, meaning that the optimized control corresponds exactly to the nominal control.

When a bigger disturbance occurs, a greater input force is again required to remain within the limits. Once more, it is evident that a value of 10 N is within the maximum force limits.

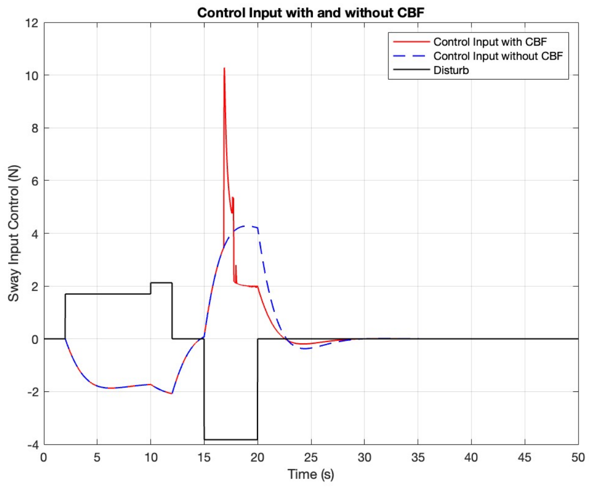

Figure 7.36: Simulation 3: Comparison of the control input obtained with and without CBF.

Figures 7.37 show the evolution of the barrier function over time. It can be observed that, between 2 and 9 seconds, the function is moderately activated but does not reach the highest value, due to the fact that a lateral displacement is present, which causes the system to approach the boundary at 0.2 m, but it does not reach the limit bound.

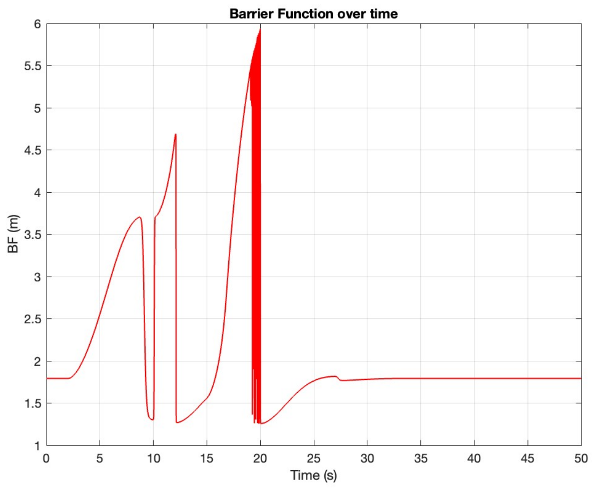

Figure 7.37: Simulation 3: Barrier Function behavior.

At $t = 10$, following the introduction of a new disturbance, the barrier function activates again, though without reaching its maximum value, as the system does not hit the 0.2 m boundary. It is important to note that the behavior of the barrier function from $t = 0$ to $t = 15$ does not significantly impact the control, as the two control inputs remain identical.

From $t = 15$ onwards, the barrier function increases once again due to the third disturbance. Around $t = 18$, a chattering phase begins. In fact, as shown in Figure 7.34, the system reaches the boundary precisely at $t = 18$ and remains there until $t = 20$, when the disturbance returns to zero, which leads the barrier function to stabilize at a constant value and bringing the two control inputs back to be equals.

Therefore, it can be concluded that the barrier function, as defined, shows a good response in the context of a lane-keeping problem, ensuring that the system remains within a predefined safe corridor. This result highlights the correct result of satisfying the constraints without requiring excessive control effort. However, more study on the selection of the approximation of the *sigmoid* function is necessary to further refine the adaptability of the method.

Chapter 8

Conclusion

The following section presents the overall conclusions. Specifically, Section 8.1 discusses the outcomes of this thesis, while Section 8.2 outlines potential directions for future research.

8.1 Thesis Outcome

This thesis presents a study on the motion planning and control of an Autonomous Underwater Vehicle (AUV), specifically the BlueROV2, for dynamic target tracking and safe trajectory execution. The research is structured into two major phases. In the first phase, a standard autopilot problem is conducted within the Gazebo simulation environment, concentrating on the surge and yaw dynamics of the AUV. The second phase advances upon this by introducing a control strategy to ensure safe path following, focusing on the sway dynamics through the incorporation of Control Barrier Functions (CBF). This method improves the AUV's ability to follow a safe path.

Initially, a mathematical modelling of the AUV was presented, followed by a simplified model focused on horizontal motion (surge, sway, yaw). This simplification allowed the derivation of an effective autopilot strategy, based on two PID controllers for surge and yaw. The implementation in the *ROS/Gazebo* environment provided a realistic simulation framework to validate the proposed approach.

A key point in the study was the need for real-time motion planning, since the AUV is required to track a moving target while avoiding obstacles. To address this, the *Rapidly-Exploring Random Tree Star* (RRT*) algorithm was selected. This algorithm was chosen because it is suitable for dynamically changing environments as it iteratively refines the trajectory. For the guidance system, a *Line-of-Sight* (LOS) approach was employed, a widely used method in marine applications due to its effectiveness in directing a vehicle along a predefined path while maintaining a smooth trajectory.

Despite the satisfactory performance of the PID-based autopilot under nominal conditions, further improvements can be made to achieve more accurate path following and to ensure that the AUV remains within a ”safe” corridor during navigation. To handle this problem, an approach based on *Control Barrier Functions* (CBFs) was coducted. A lane-keeping strategy was implemented, introducing an additional LQR controller in the sway dynamics. This modification ensured that the AUV remained within a predefined safety corridor. Assumptions regarding disturbance types and AUV velocity profiles were made in order to simulated them. The results indicate that the proposed approach successfully maintains the vehicle within safe operational bounds.

Overall, from a computational standpoint, RRT* was found to be efficient in generating feasible paths in real-time. The LOS guidance provided accurate directional control, while the PID-based autopilot offered a simple yet effective control for path following. However, in scenarios where bounded position limitations are required when following a predefined path, using CBF-based control is essential to maintain safety constraints. One of the main challenges in implementing CBF-based control is the real-time solution of the associated QP problem. However, various experiments demonstrated that practical implementations of this approach remain feasible and promising for real-world deployment.

In conclusion, the development of the *Rapidly-Exploring Random Tree Star* (RRT*), *Line-of-Sight* (LOS) guidance systems, and a PID-based autopilot for surge and yaw control in a dynamic context has proven to be effective. The integration of these systems allow an efficient path following with a real-time path planning. Additionally, the initial studies into incorporating *Control Barrier Functions* for sway control appear promising. This approach suggests potential for improving the ability of the AUV to maintain stability and safety under varying ocean conditions. The promising results encourage continued exploration and refinement of CBF integration, aiming to achieve even higher levels of safety in underwater navigation.

8.2 Future Work

Following this initial phase of study, further analyses has to be computed explored to validate the applicability of Control Barrier Functions in different operational scenarios. In order to do it, it is important to test them under different types of disturbances. Specifically, scenarios should not be constrained to cases where the Autonomous Underwater Vehicle (AUV) travels at a constant cruise speed in surge or follows a straight-line trajectory. Instead, more complex situations where both surge and yaw dynamics are actively involved should be considered. One key aspect to investigate is the coupling between sway and yaw dynamics. Activating CBFs in sway could introduce challenges during the calculation of the lookahead point in the Line-Of-Sight (LOS) guidance. Moreover, an essential factor to take into account is the maximum input that the thrusters (T200) can sustain. If part of the power is allocated to surge motion and manouvering control through yaw, it becomes necessary to accurately adress the remaining power that CBFs can utilize to counteract the disturbances. Also, to validate the functionality of CBFs in a more realistic context, their implementation in ROS/Gazebo should be computed. This includes designing a surge and yaw autopilot based on PID control, along with an additional sway control integrating an QLR control and a CBF-based QP.

Furthermore, similarly to the study of the lane keeping problem, an Adaptive Cruise Control strategy using CBFs can be developed to dynamically regulate velocity in a dynamic environment. This approach makes sure that the vehicle adapts its speed in real-time based on the environmental factors. Indeed, simulations in a dynamic environment with both static and dynamic obstacles should be explored. In such scenarios, the combination of RRT* for feasible trajectory generation considering fixed obstacles and CBFs for real-time collision avoidance could be implemented. These improvements would enable a more robust navigation and make sure the Autonomous Underwater Interception Drone of being able to generate a real-time trajectory while ensuring obstacle avoidance of dynamic obstacles. This would guarantee the correct execution during the path-following phase, keeping the system within a desired travel corridor and preventing occurrences of collision with any obstacle around.

Appendix A

Thruster T200 Characteristics

In the graph the thrust characteristics of the Blue Robotics T200 is shown. On the x axis the ESC PWM input value and on the y axis the thrust in kgf. To convert kgf to Newton, the value must be multiplied by 9.81.

For a supply voltage of 14V, value considered for this project, the thrust reaches approximately 5 kgf, which corresponds to 50 N. To remain conservative, a maximum thrust value of 40 N is considered.

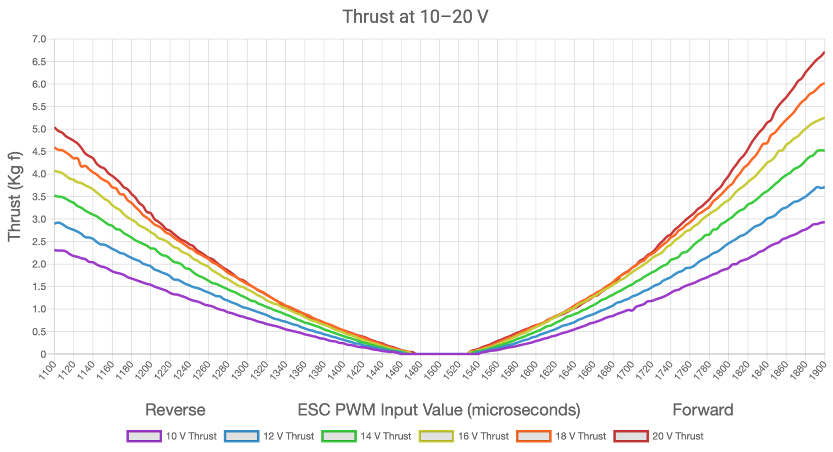

Figure A.1: Thruster T200 characteristic ESC-Thrust. [54]

Bibliography

[1] Saab AB. *Double Eagle – Underwater Vehicle*. `https://www.saab.com/products/double-eagle` (cit. on p. 2).

[2] Christian Bueger and Tobias Liebetrau. "Critical maritime infrastructure protection: What's the trouble?" In: *Marine Policy* (2023). DOI: `10.1016/j.marpol.2023.105772` (cit. on p. 1).

[3] Richard Milne, Henry Foy, and David Sheppard. "Sabotage of gas pipelines a wake-up call for Europe, officials warn". In: *Financial Times*. Online, Sept. 2022 (cit. on p. 2).

[4] *UUV Simulator*. `https://uuvsimulator.github.io` (cit. on p. 3).

[5] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Chichester, UK: Wiley, 2011. DOI: `10.1002/9781119994138` (cit. on pp. 5, 10, 14–17, 19, 27, 33–35, 39).

[6] Anastasios M. Lekkas. "Guidance and Path-Planning Systems for Autonomous Vehicles". Doctoral thesis. PhD thesis. Trondheim, Norway: Norwegian University of Science and Technology (NTNU), 2014. ISBN: 978-82-326-0177-6. URL: `https://www.researchgate.net/publication/282870336_Guidance_and_Path-Planning_Systems_for_Autonomous_Vehicles` (cit. on p. 6).

[7] Anete Vagale, Rachid Oucheikh, Robin T. Bye, Ottar L. Osen, and Thor I. Fossen. "Path planning and collision avoidance for autonomous surface vehicles: a review". In: *Journal of Marine Science and Technology* 26 (Jan. 2021), pp. 1292–1306. DOI: `10.1007/s00773-020-00787-6` (cit. on p. 6).

[8] Daoliang Li, Peng Wang, and Ling Du. "Path Planning Technologies for Autonomous Underwater Vehicles—A Review". In: *IEEE Access* 7 (2019), pp. 9745–9768. DOI: `10.1109/ACCESS.2018.2886867` (cit. on p. 7).

[9] Y. Guo, H. Liu, X. Fan, and W. Lyu. "Research Progress of Path Planning Methods for Autonomous Underwater Vehicle". In: *Mathematical Problems in Engineering* 2021 (2021), pp. 1–25. DOI: `10.1155/2021/8847863` (cit. on p. 7).

[10] Wei Zhang, Naixin Wang, and Wenhua Wu. "A hybrid path planning algorithm considering AUV dynamic constraints based on improved A* algorithm and APF algorithm". In: *Ocean Engineering* 285 (2023), p. 115333. DOI: `10.1016/j.oceaneng.2023.115333` (cit. on p. 7).

[11] Tengbin Zhu, Yingjie Xiao, and Hao Zhang. “Path planning of USV based on improved PRM under the influence of ocean current”. In: *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment* 238.4 (2024), pp. 707–1008. DOI: 10.1177/14750902231214585 (cit. on p. 7).

[12] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894. DOI: 10.1177/0278364911406761 (cit. on pp. 7, 24, 25).

[13] Simon Williams, Xuezhi Wang, Daniel Angley, Christopher Gilliam, Bill Moran, Richard Ellem, Trevor Jackson, and Amanda Bessell. “Dynamic Target Driven Trajectory Planning using RRT*”. In: *22nd International Conference on Information Fusion*. Ottawa, Canada: ISIF, 2019. DOI: 10.1109/FUSION.2019.8859023 (cit. on pp. 7, 24).

[14] Xuezhi Wang, Simon Williams, Daniel Angley, Christopher Gilliam, Trevor Jackson, Richard Ellem, Amanda Bessell, and Bill Moran. “RRT* Trajectory Scheduling Using Angles-Only Measurements for AUV Recovery”. In: *22nd International Conference on Information Fusion*. Ottawa, Canada: ISIF, 2019 (cit. on p. 7).

[15] A. Shneydor. *Missile Guidance and Pursuit: Kinematics, Dynamics and Control*. City, Country: Publisher Name, 1998. ISBN: ISBN Number (cit. on p. 7).

[16] Morten Breivik and Thor I. Fossen. “Guidance Laws for Autonomous Underwater Vehicles”. In: *INTECH Education and Publishing*. INTECH Education and Publishing, 2009. Chap. 4, pp. 51–76 (cit. on pp. 7, 8, 33).

[17] Morten Breivik and Thor I. Fossen. “Guidance-Based Path Following for Autonomous Underwater Vehicles”. In: *IEEE Journal of Oceanic Engineering* 31.1 (2006), pp. 282–291. DOI: 10.1109/JOE.2006.872370 (cit. on p. 7).

[18] Guoli Shen, Zhongjing Zhou, Cuicui Xia, Xiaoting Xu, Bo He, and Yue Shen. “Path tracking of AUV based on improved line-of-sight method”. In: *IEEE OCEANS 2021* (2022). DOI: 10.1109/OCEANS1718295.2022.9777299 (cit. on p. 7).

[19] A. M. Lekkas and T. I. Fossen. “A time-varying lookahead distance guidance law for path following”. In: *Proceedings of the 9th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)*. Arenzano, Italy, 2012 (cit. on p. 8).

[20] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. “Control Barrier Functions: Theory and Applications”. In: *2019 18th European Control Conference (ECC)*. Napoli, Italy: IEEE, 2019, pp. 3420–3425. DOI: 10.23919/ECC.2019.8796030 (cit. on pp. 8, 40, 41, 45).

[21] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. “Control Barrier Function Based Quadratic Programs for Safety Critical Systems”. In: *arXiv preprint* 1609.06408v2 (Dec. 2016). DOI: `10.48550/arXiv.1609.06408` (cit. on pp. 8, 43, 44).

[22] Syunsuke Fukuda, Yasuyuki Satoh, and Osamu Sakata. “Trajectory-Tracking Control Considering Obstacle Avoidance by using Control Barrier Function”. In: *2020 International Automatic Control Conference (CACS)*. Hsinchu, Taiwan: IEEE, 2020, pp. 1–6. DOI: `10.1109/CACS50047.2020.9289780` (cit. on p. 8).

[23] Zhigang Deng, Mohammed Tousif Zaman, and Zhenzhong Chu. “Collision avoidance with control barrier function for target tracking of an unmanned underwater vehicle”. In: *Underwater Technology* 37.1 (2020), pp. 3–11. DOI: `10.3723/ut.37.003` (cit. on p. 8).

[24] Chenggang Wang, Wenbin Yu, Shanying Zhu, Lei Song, and Xinping Guan. “Safety-Critical Trajectory Generation and Tracking Control of Autonomous Underwater Vehicles”. In: *IEEE Journal of Oceanic Engineering* 48.1 (2023), pp. 93–105. DOI: `10.1109/JOE.2022.3190635` (cit. on p. 8).

[25] Aaron D. Ames, Jessy W. Grizzle, and Paulo Tabuada. “Control Barrier Function Based Quadratic Programs with Application to Adaptive Cruise Control”. In: *53rd IEEE Conference on Decision and Control (CDC)*. Los Angeles, California, USA: IEEE, 2014, pp. 4548–4555 (cit. on p. 8).

[26] Aakar Mehra, Wen-Loong Ma, Forrest Berg, Paulo Tabuada, Jessy W. Grizzle, and Aaron D. Ames. “Adaptive Cruise Control: Experimental Validation of Advanced Controllers on Scale-Model Cars”. In: *2015 American Control Conference (ACC)*. Palmer House Hilton, Chicago, IL, USA: IEEE, 2015, pp. 1414–1421 (cit. on p. 8).

[27] Xiangru Xu, Jessy W. Grizzle, Paulo Tabuada, and Aaron D. Ames. “Correctness Guarantees for the Composition of Lane Keeping and Adaptive Cruise Control”. In: *IEEE Transactions on Automation Science and Engineering* 15.3 (2018), pp. 1216–1229. DOI: `10.1109/TASE.2018.2791601` (cit. on pp. 8, 44).

[28] Eric J. Rossetter and J. Christian Gerdes. “Lyapunov Based Performance Guarantees for the Potential Field Lane-keeping Assistance System”. In: *Journal of Dynamic Systems, Measurement, and Control* 125.4 (2003), pp. 510–522. DOI: `10.1115/1.2192831` (cit. on p. 8).

[29] Guang Yang, Bee Vang, Zachary Serlin, Calin Belta, and Roberto Tron. “Sampling-based Motion Planning via Control Barrier Functions”. In: *Proceedings of the ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)* (2020). DOI: `10.1145/3365265.3386282` (cit. on p. 9).

[30] Ahmad Ahmad, Calin Belta, and Roberto Tron. "Adaptive Sampling-based Motion Planning with Control Barrier Functions". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. Cancún, Mexico: IEEE, 2022. DOI: `10.1109/CDC51059.2022.9992378` (cit. on p. 9).

[31] Guang Yang, Mingyu Cai, Ahmad Ahmad, Amanda Prorok, Roberto Tron, and Calin Belta. "LQR-CBF-RRT*: Safe and Optimal Motion Planning". In: *arXiv preprint* 2304.00790v4 (2023). arXiv: `2304.00790 [cs.RO]`. URL: `https://arxiv.org/abs/2304.00790` (cit. on p. 9).

[32] Blue Robotics Inc. *BlueROV2 - Affordable and Capable Underwater ROV*. Accessed: 2025-01-25. 2025. URL: `https://bluerobotics.com/store/rov/bluerov2/` (cit. on p. 10).

[33] The Society of Naval Architects and Marine Engineers (SNAME). *Nomenclature for treating motion of a submerged body through a fluid*. Tech. rep. New York, USA: Technical and Research Bulletin No.1-5, 1950 (cit. on p. 11).

[34] Bingheng Wang, Marko Mihaelec, Yongbin Gong, Dario Pompili, and Jingang Yi. "Disturbance Observer-Based Motion Control of Small Autonomous Underwater Vehicles". In: *Proceedings of the ASME 2018 Dynamic Systems and Control Conference (DSCC2018)*. Atlanta, Georgia, USA: ASME, Sept. 2018, pp. 1–8. DOI: `10.1115/DSCC2018-9200` (cit. on p. 13).

[35] Andreas Baldur Nørregård Hansen. "System Identification and Dynamic Positioning of Autonomous Underwater Vehicle". MA thesis. Denmark: Technical University of Denmark (DTU), Oct. 2016 (cit. on p. 15).

[36] M. von Benzon, F. F. Sørensen, E. Uth, J. Jouffroy, J. Liniger, and S. Pedersen. "An Open-Source Benchmark Simulator: Control of a BlueROV2 Underwater Robot". In: *Journal of Marine Science and Engineering* 10.1898 (2022). DOI: `10.3390/jmse10121898`. URL: `https://doi.org/10.3390/jmse10121898` (cit. on p. 20).

[37] Chu-Jou Wu. "6-DoF Modelling and Control of a Remotely Operated Vehicle". MA thesis. Australia: Flinders University, July 2018 (cit. on p. 21).

[38] Josué González-García, Néstor Alejandro Narcizo-Nuñ, Luis Govinda Garcia-Valdovinos, Tomás Salgado-Jiménez, Alfonso Gómez-Espinosa, Enrique Cuan-Urquizo, and Jesús Arturo Escobedo Cabello. "Model-Free High Order Sliding Mode Control with Finite-Time Tracking for Unmanned Underwater Vehicles". In: *Applied Sciences*. Vol. 11. 4. 2021, p. 1863. DOI: `10.3390/app11041863` (cit. on p. 21).

[39] Ali Arifi, Julien Lepagnot, Soufiene Bouallègue, and Laetitia Jourdan. "3D Path Planning of Autonomous Underwater Vehicles Using a Rapidly-exploring Random Trees Algorithm". In: *Proceedings of the 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (2023), pp. 1–6. DOI: `10.1109/ICSMC55871.2023.10364590` (cit. on p. 24).

[40] James J. Kuffner Jr. and Steven M. LaValle. “RRT-Connect: An Efficient Approach to Single-Query Path Planning”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2000, pp. 995–1001. DOI: `10.1109/ROBOT.2000.844730` (cit. on p. 25).

[41] Lester E. Dubins. “On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents”. In: *American Journal of Mathematics* 79.3 (1957), pp. 497–516. DOI: `10.2307/2372560` (cit. on p. 27).

[42] Lijun Yu, Zhihong Wei, Zhengan Wang, Yukun Hu, and Hui Wang. “Path Optimization of AUV Based on Smooth-RRT Algorithm”. In: *Proceedings of the IEEE International Conference on Automation, Robotics and Applications* (2025), pp. 1–8 (cit. on p. 28).

[43] Yiyang Liu, Yang Zhao, Shuaihua Yan, Chunhe Song, and Fei Li. “A Sampling-Based Algorithm with the Metropolis Acceptance Criterion for Robot Motion Planning”. In: *Sensors* 22.23 (2022), p. 9203. DOI: `10.3390/s22239203`. URL: `https://doi.org/10.3390/s22239203` (cit. on p. 29).

[44] Anastasios M. Lekkas and Thor I. Fossen. “Line-of-Sight Guidance for Path Following of Marine Vehicles”. In: *Advanced in Marine Robotics*. Ed. by Oren Gal. Lambert Academic Publishing, June 2013. Chap. 5. URL: `https://www.researchgate.net/publication/` (cit. on pp. 31, 32).

[45] N. Minorsky. *Directional Stability of Automatically Steered Bodies*. Tech. Res. Bull. 2. Journal of the American Society for Naval Engineers, 1922, pp. 280–309. DOI: `10.1111/j.1559-3584.1922.tb04958.x` (cit. on p. 35).

[46] M.R. Katebi, M.J. Grimble, and J. Byrne. *LQG Adaptive Autopilot Design*. Tech. Res. Bull. Volume 18, Issue 5, Pages 1293-1298. IFAC Proceedings Volumes, 1985. DOI: `10.1016/S1474-6670(17)60742-0` (cit. on p. 35).

[47] B. Jalving. “The NDRE-AUV flight control system”. In: *IEEE Journal of Oceanic Engineering* 19.4 (2002), pp. 497–501. DOI: `10.1109/48.338385` (cit. on p. 35).

[48] K. Nomoto, T. Taguchi, K. Honda, and S. Hirano. “On the Steering Qualities of Ships”. In: *International Shipbuilding Progress* 4.35 (1957) (cit. on p. 37).

[49] Xiangru Xu, Thomas Waters, Daniel Pickem, Paul Glotfelter, Magnus Egerstedt, Paulo Tabuada, Jessy W. Grizzle, and Aaron D. Ames. “Realizing Simultaneous Lane Keeping and Adaptive Speed Regulation on Accessible Mobile Robot Testbeds”. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)* (2017), pp. 1762–1768. DOI: `10.1109/CCTA.2017.8062758` (cit. on p. 45).

[50] Wei Xiao and Calin Belta. “High-Order Control Barrier Functions”. In: *IEEE Transactions on Automatic Control* 67.7 (2022), pp. 3655–3662 (cit. on p. 47).

[51] Musa Morena Marcusso Manhães, Sebastian A. Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. "UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation". In: *OCEANS 2016 MTS/IEEE Monterey*. IEEE, 2016, pp. 1–8. DOI: `10.1109/OCEANS.2016.7761080` (cit. on p. 49).

[52] Damitha Sandaruwan, Nihal Kodikara, Rexy Rosa, and Chamath Keppitiyagama. "Modeling and Simulation of Environmental Disturbances for Six degrees of Freedom Ocean Surface Vehicle". In: *Sri Lankan Journal of Physics* 10 (2009), pp. 39–57 (cit. on p. 59).

[53] Monika Bortnowska. "Prediction of power demand for ship motion control system of sea mining ship fitted with tubular winning system". In: *Polish Maritime Research* 4(54).Vol 14 (2007), pp. 24–30. DOI: `10.2478/v10012-007-0036-7` (cit. on p. 59).

[54] Blue Robotics. *T200 Thruster*. 2025. URL: `https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/` (cit. on p. 73).