FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Control for Docking of Autonomous Surface Vehicle

Diogo Araújo de Oliveira



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Bruno Ferreira

November 5, 2025

© Diogo Oliveira, 2025

Resumo

Os Veículos Autónomos de Superfície (ASVs) são veículos concebidos para operações marítimas como recolha de dados, reabastecimento e monitorização ambiental. Mas atracar um ASV é uma tarefa desafiante porque os ambientes marinhos são muito dinâmicos e imprevisíveis devido às perturbações muito comuns de ondas, correntes e obstáculos. Este trabalho aborda estes desafios através da exploração e implementação de um Modelo Preditivo de Controlo (MPC) optimizada para atracagem de ASVs.

O MPC é uma técnica de controlo emergente capaz de lidar com restrições multivariáveis e optimização de sistemas em tempo real.

O foco deste trabalho envolve a concepção e implementação de um modelo matemático de um ASV real, controlado por um algoritmo MPC também concebido e implementado. Para testar o sistema, foi entendido que era necessário um cenário mais complexo, pelo que, embora não tenha sido inicialmente planeado, foi também desenvolvido um sistema de orientação bem como um planeador de trajectória global de forma a controlar o sistema em situações mais complexas. Para simulação, foi implementado um ambiente virtual com perturbações de ondas, correntes e obstáculos dinâmicos. O sistema foi testado em 3 casos distintos com complexidade crescente , avaliando métricas como erro de posicionamento, tempo de atracagem, consumo energético e capacidade de respeitar restrições.

Os resultados demonstram que o controlador MPC desenvolvido, com a inovadora capacidade de incorporar o desvio de obstáculos, a consegue realizar manobras de atracagem com elevada precisão mesmo na presença de ruído nos sensores, respeitando sempre as restrições físicas do sistema. Este projeto visa contribuir para os avanços dos sistemas marítimos autónomos através da melhoria da precisão, robustez e eficiência operacional das manobras de atracagem de ASVs.

Abstract

Autonomous Surface Vehicles (ASVs) are vehicles designed for maritime operations such as data collection, refueling, and environmental monitoring. However, docking an ASV is a challenging task, as marine environments are highly dynamic and unpredictable due to frequent disturbances from waves, currents, and obstacles. This work addresses these challenges through the exploration and implementation of a Model Predictive Control (MPC) strategy optimized for ASV docking.

MPC is an emerging control technique capable of handling multivariable constraints and real-time system optimization.

The focus of this work involves the design and implementation of a mathematical model of a real ASV, controlled by an MPC algorithm that was also developed and implemented. To properly test the system, it was determined that a more complex scenario was required. Therefore, although not initially planned, a guidance system as well as a global trajectory planner were also developed to enable control of the system under more complex conditions. For simulation purposes, a virtual environment was implemented that included disturbances from waves, currents, and dynamic obstacles. The system was tested under three distinct scenarios of increasing complexity, evaluating metrics such as positioning error, docking time, energy consumption, and constraint compliance.

The results demonstrate that the developed MPC controller, with its innovative capability to incorporate obstacle avoidance, is able to perform docking maneuvers with high precision even in the presence of sensor noise, while consistently respecting the physical constraints of the system. This project aims to contribute to the advancement of autonomous maritime systems by improving the accuracy, robustness, and operational efficiency of ASV docking maneuvers.

Acknowledgements

A entrega desta dissertação marca o fim do meu bonito capítulo como estudante, do qual me orgulho e no qual tenho confiança que me tornou uma pessoa melhor, ou pelo menos, mais espelhado com aquilo que me visualizo a ser. Tenho então que agradecer ao curso de Engenharia Eletrotécnica e de Computadores por ter tornado tudo isto uma realidade e fazer de mim uma pessoa mais madura, consciente, interessante, interessada, curiosa e acima de tudo, feliz. Gostaria ainda de agradecer a toda a gente que ao longo destes 5 anos moldou o meu caminho. Um agradecimento especial vai para toda a minha família, mas sobretudo à minha mãe, ao meu pai e ao meu irmão, que foram incansáveis não só estes 6 meses, mas todos os 5 anos desde o dia que me viram partir para outra cidade com saudade até hoje, em que termino este ciclo, do qual eles terão também sentido cada dificuldade, fazendo sempre de tudo para me ajudar a superar. A eles dedicarei todas as minhas conquistas, e esta não é exceção.

Um agradecimento especial vai também para a minha namorada, que tornou todo este caminho muito mais especial e que, especialmente nestes últimos 6 meses meu apoiou em todos os momentos, tendo sido seguramente um poço de força para mim.

Obrigado a todos os meus amigos, desde os amigos de Barcelos, que mesmo após longos anos continuam a escolher partilhar momentos comigo, a todos os amigos que fiz no Porto (e um agradecimento muito especial a quem partilhou ambos) que têm uma importância na minha vida que nem eles próprios sabem que a têm, e acima de tudo, obrigado à SEDE que foi minha amiga em muitos momentos e que sempre me ajudou a aligeirar quaisquer momentos menos altivos na minha vida.

Agradeço também ao NEEEC-FEUP por me ter permitido fazer parte de um grupo de trabalho inesquecível em que conheci amigos para a vida e ao mesmo tempo contribuir para a atividade do curso e integração de estudantes.

Por fim, um obrigado a todos os professores que fizeram parte desta jornada, em especial ao prof. Bruno Ferreira, meu orientador, por me ter dado a oportunidade de trabalhar com ele, e que ao longo de mais de um ano me ajudou a conquistar objetivos que por vezes duvidei serem alcançáveis.

Obrigado a todos por fazerem e terem feito parte da minha vida. Serão sempre relembrados !

Diogo Oliveira

"This is the last you shall hear from me... I now walk out to live amongst the wild."

Christopher MacCandless

Contents

Acknowledgements iii

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Objectives . . . 2
- 1.3 Document Structure . . . 3

2 Background and Literature Review 5
- 2.1 Dynamics modelling for ocean vehicles . . . 5
- 2.2 Global planning . . . 6
  - 2.2.1 Path Planning . . . 6
  - 2.2.2 Path smoothing . . . 11
- 2.3 Guidance and Control . . . 12
  - 2.3.1 Guidance . . . 12
  - 2.3.2 Control . . . 14
- 2.4 Related Work . . . 18
- 2.5 Summary and Analysis . . . 20

3 Homing and Docking Control Framework for ASVs 22
- 3.1 System Architecture . . . 23
  - 3.1.1 Hierarchical System Framework . . . 23
  - 3.1.2 Real-Time GNC Loop Architecture . . . 25
  - 3.1.3 Performance and Safety Considerations . . . 26
- 3.2 Introduction to MPC . . . 27
- 3.3 Mathematical Modeling . . . 28
  - 3.3.1 Vehicle Dynamics . . . 28
  - 3.3.2 MPC Controller . . . 34
- 3.4 Global Planning . . . 40
  - 3.4.1 Map construction . . . 40
  - 3.4.2 Path Planning . . . 41
- 3.5 Guidance and Control Framework . . . 45
  - 3.5.1 Guidance System Structure . . . 45
  - 3.5.2 MPC . . . 48
  - 3.5.3 Cost Function Design . . . 52
  - 3.5.4 Simulation Control Scheme . . . 54
- 3.6 Strategies and Computational Tools . . . 55

4 Experimental Validation and Results **57**
4.1 Introduction . . . 57
4.2 Preliminary Results . . . 58
4.3 Final Simulation Environment and Results . . . 59
4.3.1 Use case PH . . . 60
4.3.2 Use case PHO . . . 62
4.3.3 Use case C . . . 64
4.3.4 Introducing Noise . . . 66
4.4 Discussion . . . 68

5 Conclusion and Future Work **70**
5.1 Conclusion . . . 70
5.2 Future Work . . . 71

References **72**

A Complete Dynamic Model Assembly **78**

List of Figures

1.1 ZARCO ASV . . . 2
1.2 Marina de Leça da Palmeira . . . 3

2.1 D* algorithm grid planning . . . 7
2.2 Iteration of the RRT algorithm . . . 8
2.3 RRT* rewiring step illustration . . . 9
2.4 Quiver plot of robot navigation using simple potential field . . . 9
2.5 Artificial Potential Field Navigation Function Slope Map . . . 10
2.6 Smoothing methods comparison with sample control points . . . 11
2.7 Simplified flow diagram for guidance and control . . . 12
2.8 Trajectory tracking vs. Path following [49] . . . 13
2.9 LOS guidance principle . . . 13
2.10 Closed Loop Control System Principals [47] . . . 14
2.11 PID Control System Diagram [4] . . . 15
2.12 MPC graph illustration . . . 16
2.13 Optimization Procedure of PSO . . . 18

3.1 System Workflow Diagram . . . 24
3.2 GNC Detailed Diagram . . . 26
3.3 Guidance, Control, Motion and Plot relations . . . 26
3.4 MPC generalized scheme . . . 28
3.5 Relation between Earth fixed frame (East-North-Down ENU) system with vehicle Body fixed frame . . . 29
3.6 Rk4 method: 4 slopes representation. . . . 38
3.7 Map construction complete pipeline . . . 42
3.8 Spline Path Visualization . . . 45
3.9 Guidance system diagram . . . 45
3.10 Visualization of power term variation in Superellipses . . . 49
3.11 Initial Map . . . 50
3.12 Densified polygon line segments for obstacle avoidance . . . 50
3.13 Simulation control scheme for defining when stop . . . 55

4.1 UI used for simulation tests . . . 57
4.2 Example of the first map developed for simulation . . . 58
4.3 Example of the second version of the map used for simulation . . . 59
4.4 Use Case PH (Heading not aligned) : Simulation . . . 60
4.5 Use Case PH (Heading not aligned) : Linear and angular velocities and $[x, y, \psi]$ changing over time. . . . 60

4.6 Use Case PH (Heading not aligned) : Thrusters input variation along time and time to solve the OCP problem . . . 61
4.7 Use Case PH (Heading aligned) : Simulation . . . 61
4.8 Use Case PH (Heading aligned) : Linear and angular velocities and $[x, y, \psi]$ changing over time. . . . 62
4.9 Use Case PH (Heading aligned) : Thrusters input variation along time and time to solve the OCP problem. . . . 62
4.10 Use Case POH : Simulation . . . 63
4.11 Use Case POH : Linear and angular velocities and $[x, y, \psi]$ changing over time. . . . 63
4.12 Use Case POH : Thrusters input variation along time and time to solve the OCP problem. . . . 63
4.13 Use Case C (heading not aligned) : Short movement for the dock, simulation . . . 64
4.14 Use Case C (heading not align) : Linear and angular velocities and $[x, y, \psi]$ changing over time . . . 64
4.15 Use Case C (heading not align): Thrusters input variation along time and time to solve the OCP problem . . . 65
4.16 Use Case C (heading aligned) : Larger movement simulation . . . 65
4.17 Use Case C (heading aligned) : Linear and angular velocities and $[x, y, \psi]$ changing over time . . . 65
4.18 Use Case C (heading aligned) : Thrusters input variation along time and time to solve the OCP problem . . . 66
4.19 Use Case PH : Simulation with noise on thruster's input. . . . 66
4.20 Use Case Use Case PH with noise : Linear and angular velocities and $[x, y, \psi]$ changing over time . . . 67
4.21 Use Case Use Case PH with noise : Thrusters input variation along time and time to solve the OCP problem . . . 67
4.22 Use Case C : Simulation with noise on thruster's input. . . . 67
4.23 Use Case Use Case C with noise : Linear and angular velocities and $[x, y, \psi]$ changing over time . . . 68
4.24 Use Case Use Case C with noise : Thrusters input variation along time and time to solve the OCP problem . . . 68

List of Acronyms

GNC	Guidance and control
MPC	Model Predictive control
NMPC	Nonlinear Model Predictive Control
ASV	Autonomous Surface Vehicle
AAV	Autonomous Aerial Vehicle
AUV	Autonomous Underwater Vehicle
PID	Proportional-Integral-Derivative
LiDAR	Light Detection and Ranging
IMU	Inertial Measuring Unit
LQR	Linear Quadratic Regulator
QP	Quadratic Programming
OCP	Optimal Control Problem
NLP	Nonlinear Programming
AI	Artificial Intelligence
NN	Neural Networks
RRT	Rapid Exploring Random Tree
APF	Artificial Potential Field
SMC	Sliding Mode Control
PRM	Probabilistic Roadmap
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
CBF	Control Barrier Function
WP	Waypoint

Chapter 1

Introduction

Over the past decade, the world has been undergoing profound alterations, especially in maritime environments. With climate shifts, coastal regions have been facing rising sea levels [5] and environmental regulations are becoming stricter, creating the need for more sustainable practices in ocean environments and continuous monitoring of marine ecosystems. Offshore infrastructures and industries are also changing [12], expanding into deeper and harsher waters, demanding 24/7 capabilities with reduced human exposure. In parallel, the very recent heightened investments in defense and the urgency for humanitarian assistance have also pressured automation in remote and dangerous environments and accelerated the need for intelligent autonomous systems.

At the same time, advancements in sensing, computing, and robotics have opened new possibilities for how the systems interact with and monitor the environment, setting the stage for Autonomous Surface Vehicles (ASVs) to be a pivotal technology for this transformation.

ASVs are crucial to perform tasks like water pollution monitoring, ocean data collection, remote infrastructure inspections [7], serve defense and surveillance missions, or even serve as communication relays as these are often in cooperation with other vehicles such as autonomous underwater vehicles (AUVs) [59] or unmanned aerial vehicles (AAVs). All achieved without the need for human intervention, reducing risk and complexity, and this is possible due to their ability to operate continuously, for very extended periods, in often complex environments, keeping flexibility and precision.

Among the many challenges that ASVs face to complete such missions, docking stands out as a critical one, being transversal to all types of operations.

Docking is a high-precision task that requires the ASV to navigate in close proximity to static structures under the influence of dynamic environmental forces such as wind, currents, and waves.

For these reasons, research on this topic has been growing in recent years, with big interest in intelligent systems capable of adapting to changing environments and constrained, complex scenarios.

1.1 Motivation

Docking is a high-precision task that requires the ASV to navigate in close proximity to static structures under the influence of dynamic environmental forces such as wind, currents, and waves.

Autonomous docking is a key enabler of long-term ASV operations, since it allows critical operations such as battery recharging, data offloading, and equipment transfer without human intervention. These capabilities are essential for the transition from supervised to fully autonomous maritime operations.

In practical terms, the success of autonomous maritime missions often is defined on whether docking can be performed accurately and safely under uncertain and dynamic conditions, or not, as ASVs are expected to operate in increasingly demanding scenarios.

However, docking remains one of the most complex operations for marine robots due to the combined effects of unpredictable ocean conditions and the need for fine-tuned control near physical obstacles. Failing to dock accurately can result in mission failure, equipment damage, or even environmental harm.

Achieving robust docking could leverage future development in maritime missions, paving the path for further research on this, and all the interconnected topics.

1.2 Objectives

Ensuring robust docking is not only vital for mission continuity, but also for enabling long-duration, multi-stage operations, reducing reliance on human intervention.

This project seeks to explore the existing strategies and methods for ASV docking, and develop solutions that tackle the unique challenges of marine operations and autonomous systems with the main focus being on guidance and control (GNC), as these are the most critical section of these operations. There were also taken into account algorithms that account for environmental disturbances such as ASV dynamics, integrated within the system, static obstacle avoidance, and, in the end tests were conducted to evaluate the performance across various scenarios. This project is intended to be used in INESCTEC's ZARCO 1.1 ASV.



Figure 1.1: ZARCO ASV

For validation,it was used a near-real simulated model. The docking is set to be made in 'Marina de Leça da Palmeira' 1.2 where ZARCO is currently standing, so a near-real representation of this harbor was also developed in order to achieve a realistic simulation.

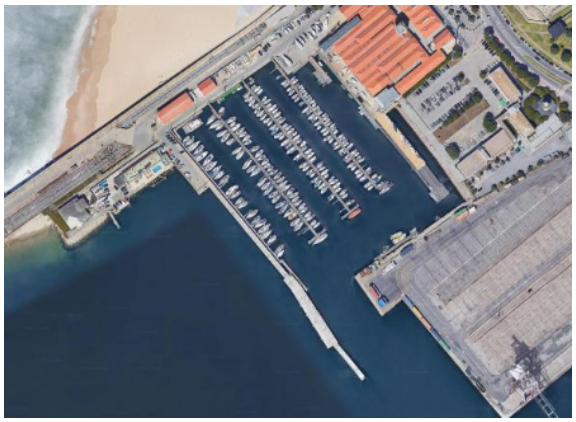


Figure 1.2: Marina de Leça da Palmeira

This dissertation addresses the pressing need for reliable and autonomous docking strategies by exploring control strategies and methods that can operate effectively amidst environmental uncertainty, system constraints, and limited onboard computational resources. By focusing on this critical capability, this work aims to serve as a foundation for further advancements in intelligent maritime autonomy and to contribute to the broader development of fully autonomous ASV systems.

1.3 Document Structure

In addition to the introduction, this dissertation contains 5 more chapters.

Chapter 2 presents the state of the art of this field and validates it with a concise literature review, from where different ideas for the development of the project were taken. The different approaches were also discussed in order to understand which methodologies could result in a better solution with exciting and robust results.

In the third chapter 3, all the methodologies used and developed to achieve the final solution were exposed and explained. It starts by giving a context on the overall system framework and information flow, and then it proceeds to expand on all the mentioned topics. From mathematical modeling to higher layer systems such as map building and global planning to the lower level systems, Guidance and Control, which are the core of this work. It is also mentioned how the simulation was created for validation and the tools used for every process ?? of the project.

Once explained how the system works, chapter 4 provides validation of the work through simulation results and metrics, all of which are explained.

Finally, chapter 5 provides a brief conclusion on the achieved results. It mentions the positives and negatives aspects of the work and concludes by mentioning at could be improved, how it could be improved and other possible features that a future work could add.

Chapter 2

Background and Literature Review

Docking is a critical operation in marine environments, as it serves as the transition point for all kinds of resources between the sea and the land. It is a complex maneuver that requires precise control and extra safety due to tight spaces in the berth and the presence of various obstacles, such as piers, other vessels, and mooring equipment.

Environmental disturbances such as wind, currents and waves, and flow patterns like tidal variation can also significantly impact the ASV's stability and docking precision requiring advanced guidance and control systems to ensure accurate positioning and avoid collisions.

Unlike traditional vessels, ASVs rely entirely on onboard algorithms for control, making precise and reliable docking highly dependent on the performance of the control system.

The process usually includes using sensor data in real-time, like LIDAR, sonar, and cameras, to find the berth and environment nearby. Along with algorithms for planning paths and avoiding obstacles, the ASV changes its movements to match the docking target. Being precise is very important to guarantee smooth transfer operations, whether it is about loading or unloading cargo, moving scientific equipment, or recharging ASV batteries. For this, real-time computation is needed.

This chapter aims to provide an essential foundation for understanding the context, challenges, and technical framework surrounding this research. Beginning with the analysis of what autonomous surface vehicles are, how they are modeled, and what their main applications are in the diverse real-world scenarios. Then a brief introduction

2.1 Dynamics modelling for ocean vehicles

The dynamic behavior and control strategies for marine vessels are more complex than those in terrestrial environments, primarily due to the continuous influence of external forces such as waves, wind, and currents, significantly influencing the control strategies for ASVs. In reality, ships and marine vehicles operate along six degrees of freedom (DOF), including surge, sway, heave, roll, pitch, and yaw. The motion control of marine vehicles is considered, with an emphasis on the nonlinear dynamics and environmental disturbances.

Marine dynamics involve modeling the forces and moments acting on a marine vehicle, such as a ship or underwater vehicle, to predict its motion. These dynamics are described by a set of nonlinear differential equations that account for hydrodynamic forces, added mass, damping, and control inputs (e.g., thrusters). Guidance and Control of Ocean Vehicles by Thor Fossen (1994) [16] provides an essential framework for understanding these dynamics and their implications on control systems. Fossen's work emphasizes the role of hydrodynamic effects such as added mass, damping and restoring forces. These factors create challenges in achieving precise control, especially in docking, where high stability is essential for the ASVs safety and performance. Today, there is a widely accepted equation (2.1) to model the dynamics of marine vehicles. However, even though a "consensual" equation exists at a starting point, ocean vehicle dynamics are complex and adaptations are often needed to accurately represent the behavior of each specific vehicle.

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau \tag{2.1}$$

- M: inertia matrix including the added mass.
- ν: vector of velocities in the body-fixed frame.
- $C(\nu)$: coriolis and centripetal matrix.
- $D(\nu)$: damping matrix.
- $g(\eta)$: gravitational and buoyancy forces.
- τ: vector of control inputs (forces and moments).

Recently, neural network-based approaches have also been employed to approximate marine vehicle dynamics[24], offering the capability to capture complex nonlinear behaviors that conventional physics-based models may oversimplify. .

2.2 Global planning

Global Planning is a crucial step for point-to-point navigation in obstacle dense environments. It involves the formulation of optimal trajectories and paths ASVs from a start to a goal position, considering the presence of static obstacles within the operational marine environment. Approaches to global planning can be essentially classified into four categories: graph-based, sampling-based, potential field, and artificial intelligence (AI)-based methods. Each category presents unique advantages and limitations that must be considered depending on the operational scenario.

2.2.1 Path Planning

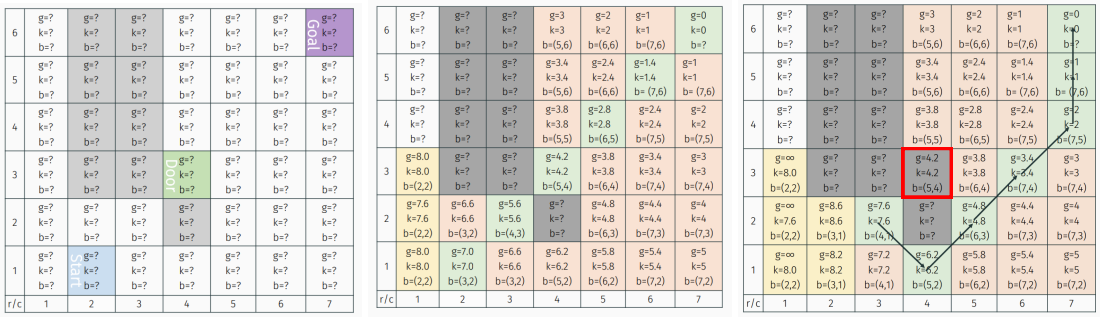
Graph-Based Approaches

Graph-based path planning methods discretize the navigation environment into grids, lattices, or interconnected nodes, forming graphs that represent feasible states and possible transitions.

Among the earliest and most influential graph search methods is Dijkstra's algorithm [11], which systematically explores paths from a start node to all reachable nodes, guaranteeing optimal solutions by consistently selecting nodes with the lowest cumulative cost. Although robust and optimal, Dijkstra's approach explores paths uniformly in all directions, potentially causing unnecessary computational overhead, particularly in large environments.

To address this inefficiency, the A* algorithm, introduced by Hart et al. [23], enhances Dijkstra's method by incorporating heuristic guidance. A* evaluates nodes based on the function $f(n) = g(n) + h(n)$, where $g(n)$ is the cumulative cost from the start node to node n, and $h(n)$ is a heuristic estimate of the remaining cost to reach the goal. This heuristic-driven approach significantly accelerates convergence to the optimal path, particularly in structured environments.

However, the fundamental assumption of static environments limits A*'s effectiveness in highly dynamic and uncertain maritime settings. This limitation has led to the development of dynamic variants, such as D* (Dynamic A*) 2.1 proposed by Stentz [52], and the Anytime Repairing A* (ARA*) introduced by Koenig and Likhachev [34], which are specifically designed for real-time replanning under evolving environmental conditions.



(a) Initial map. (b) Best path with **door opened**. (c) Best path with **closed door**.

Figure 2.1: D* algorithm grid planning. Each cell displays three values: the current cost estimate from the start node ***g()***, the heuristic estimate to the goal node ***k()***, and the backpointer ***b()***, indicating the previous cell from which the current cell was reached.[1]

Unlike static algorithms such as A*, D* continually recalculates paths whenever new obstacles (gray cells) are encountered or known paths become blocked. The path highlighted by the bold black line represents the optimal route recalculated in response to newly detected obstacles. This adaptive feature allows D* to efficiently and robustly guide navigation in environments with changing conditions and uncertain information.

Sampling-Based Approaches

Sampling-based planning algorithms address motion planning problems by avoiding the discretization of the configuration space. Instead, they rely on random sampling to incrementally explore

[1] Images Source: University of Zagreb's Faculty of Electrical Engineering and Computing (FER) slides from the 'Autonomous Mobile Robots' course.

feasible paths from start to goal. This approach is particularly advantageous in high-dimensional and complex environments.

One of the earliest and most well-known methods in this family is the Probabilistic Roadmap (PRM), introduced by Kavraki et al. [29]. PRMs operate in two phases: a preprocessing phase, where a roadmap is built by randomly sampling collision-free configurations and connecting them using local planners; and a query phase, where the start and goal positions are connected to the roadmap to extract a valid path. This could make them suitable for ASV operations in structured or semi-structured maritime spaces such as ports or offshore facilities. However, PRMs typically require prior knowledge of the environment and struggle with real-time adaptability in dynamic contexts.

In contrast, Rapidly-exploring Random Trees (RRT), introduced by LaValle [35], are single-query planners that grow a tree rooted at the start configuration by iteratively sampling the space and connecting the sample to the nearest existing node in the tree 2.2. The RRT excels at exploring large and high-dimensional configuration spaces, making it a natural fit for ASV missions in cluttered or partially known environments. Its exploratory bias enables it to quickly find feasible paths, even in non-convex domains. However, RRT does not guarantee path optimality and often produces jagged or inefficient trajectories.

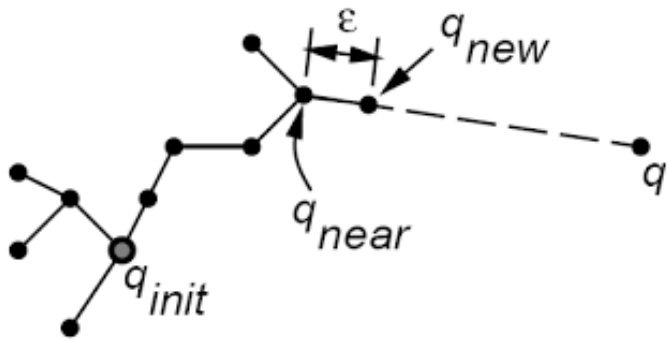


Figure 2.2: One iteration of the RRT algorithm, where ***qnew*** becomes the next node in the tree.[2]

To overcome these limitations, RRT* was proposed by Karaman and Frazzoli [28], extending RRT with an rewiring procedure. Instead of simply connecting a new sample to its nearest neighbor, RRT* evaluates a neighborhood of nearby nodes to minimize the total path cost to each node, not just the immediate edge cost to the nearest neighbor2.3. This allows the tree to evolve over time toward increasingly optimal solutions. As the number of samples increases, RRT* guarantees convergence to the optimal path, making it a robust choice for time-sensitive missions with performance constraints.

[2]Image source: Carnegie Mellon University: Link to slides.

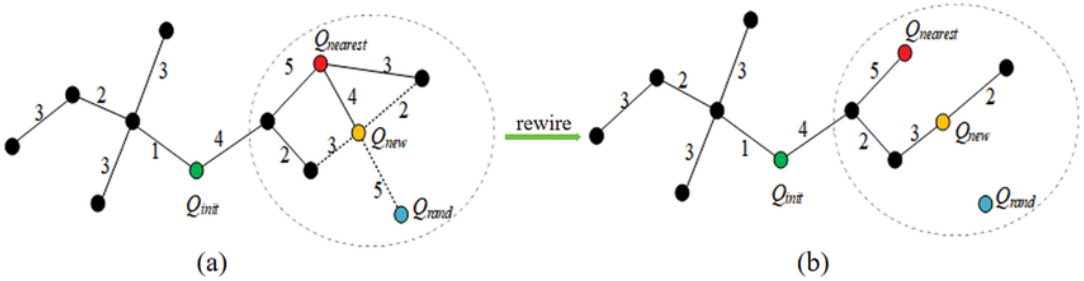


Figure 2.3: RRT* rewiring step illustration[26]

.

Beyond their classical formulations, both PRM and RRT families have been extended to account for dynamic constraints, sensor uncertainty, and dynamic obstacle avoidance, enabling their use in real-time ASV deployments. Variants such as Kinodynamic RRT[62] or PRM* incorporate motion dynamics or probabilistic guarantees, making them suitable for more realistic modeling of marine vehicles with bounded control inputs and nonholonomic constraints. Methods trying to accelerate the convergence of these algorithms, such as Bidirectional RRT* or approaches combining the strength of sampling-based planners with the power of optimal planning (2.2.1) are also being a big topic of research in recent years [63, 64].

Artificial Potential Field and AI-Based Approaches

Another variant of path planning algorithms are Artificial Potential Field (APF) planners. First introduced by in 1986 by Khatib [32], these simulate a artificial field through attractive and repulsive forces that guide the robot towards the desired goal. In these methods, goal exerts a attractive force while obstacles exert repulsive forces. The robot follows the gradient descent of this field to reach the target while avoiding obstacles 2.4.

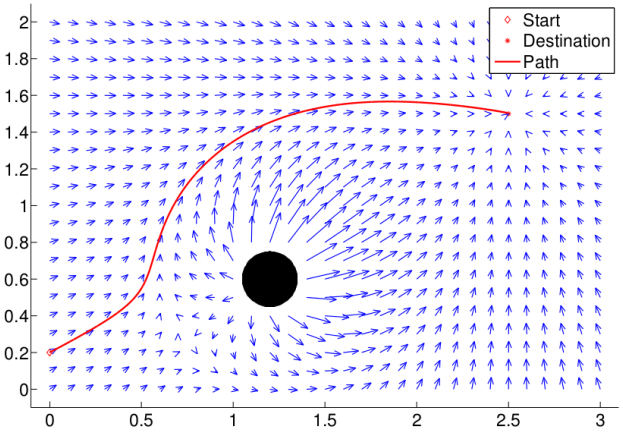


Figure 2.4: Quiver plot of robot navigation using simple potential field. In Red, the robot path from start (bottom left) to end (upper right) position. Blue arrows show the potential field, repulsive around the obstacle and attractive towards goal [3]

These are very useful due to its ease of implementation but has a big limitation which is that it may fall in local minima for convex problems, not achieving the most optimized solution.

Hao et al, suggest a improvement on the APF method to reduce its limitations [22] but adaptations like that are often complex and may require extra computational power.

The most common approach to tackle this problem are Navigation Functions 2.5. Introduced by Rimon and Koditschek in 1992, [48] Navigation Functions suggest a local minima free solution for artificial potential fields, defined by a function which has an unique minimum at goal and max value near obstacle boundaries 2.2.

$$\phi(x) = \frac{\|x - x_g\|^2}{\left(\|x - x_g\|^{2k} + \beta(x)\right)^{1/k}} \tag{2.2}$$

Here, x is the robot's position, x_g is the goal position, $\beta(x)$ is a function representing the influence of obstacles, and k is a tuning parameter. The resulting is a scalar field that guides the robot through gradient descent toward the goal while avoiding obstacles as seen in the figure 2.5.

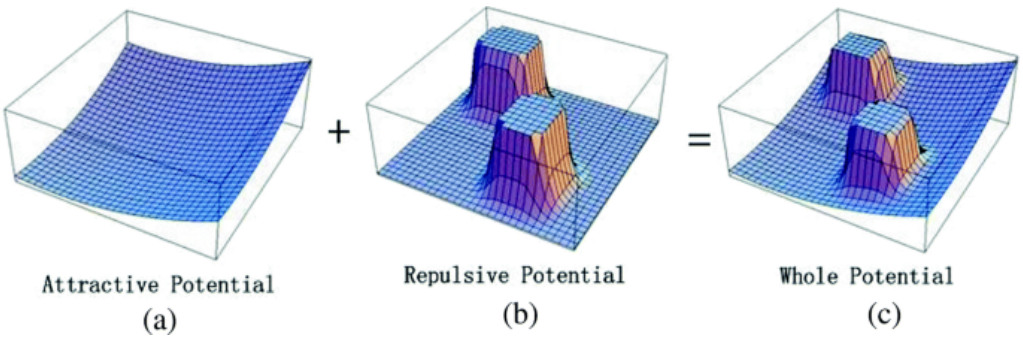


Figure 2.5: APF Navigation Function Slope Map [4]

This method has further been applied in modern solutions, although it has the limitation of bounding the map as a circle, which makes their application harder as it becomes complex to generalize.

Logunov et al. [39] extends APF method by combining polygon extraction techniques with adaptive repulsive force assignment, adding semantic context to obstacles. In marine navigation scenarios, for example, obstacles like buoys may generate less repulsive influence compared to structures like walls due to its less risk associated semantic.

The research on AI approaches have been also increasing, for example, in [54] an approach using Deep Reinforcement Learning is tested for path planning. The results obtained are interesting but the number of training iterations may have a big variation depending on the initial pose and the tuning parameters.

[3] Image source: Boston University Website.

[4] Image source: Carnegie Mellon University, Qatar branch - Link to slides.

2.2.2 Path smoothing

Path finding algorithms often deliver jagged linear paths , which are not optimal, due to very tight and regular changes of direction. Although these paths are feasible to avoid obstacles, they are generally not optimal for smooth control execution. Rough paths can lead to unstable control commands, higher actuator stress and poor precision.

To mitigate these problems, smoothing techniques often using interpolation or approximation curves, are often applied as a post-processing step, after the path finding algorithm. These techniques, refine the original trajectory, making them feasible and smooth for control.

Among the most common methods are spline-based approaches, heuristic-based smoothing, and optimization-driven techniques.

Spline approaches such as Bézier curves, introduced by Pierre Bézier [3] and B-splines, introduced by Schoenberg [49] in 1946, use approximation curves to smooth the reference trajectory and generate smooth trajectories with continuous curvature profiles, suitable for efficient and stable control, although, due to their nature, these do not guarantee that the curve will pass through all the trajectory points. On the other hand, cubic spline, formalized by Ahlberg et al. [1], are specific types of interpolation curves, which although possibly not as smooth, these curves always pass through all the trajectory points, making them useful for tight, very constrained scenarios.

Figure 2.6 shows a comparison of some of the mentioned smoothing techniques with sample control points, created for the purpose of the comparison.

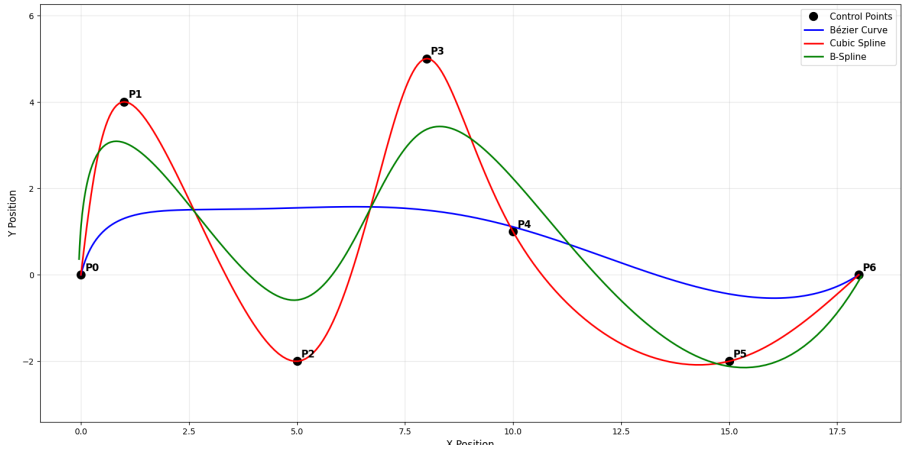


Figure 2.6: Smoothing methods comparison with sample control points

Path finding and post-processing with smoothing algorithms is often used in robotics and make a crucial combination for precise control for all types of robots that require motion, wether these are vehicles [50] or robotic arms in 2 or 3 dimensional space.

After getting a smoothed path, feasible for navigation, it's necessary to use them in control, in order to obtain the desired behavior from the robot.

2.3 Guidance and Control

Marine vehicle control strategies can be broadly categorized into guidance and control. Guidance involves determining a desired path or set of waypoints for the ASV to follow while control focuses on making real-time adjustments to keep the ASV with the desired behavior 2.7. Control strategies must adapt to external disturbances and uncertainties, particularly in docking scenarios where proximity to other vessels or structures requires extremely accurate maneuvering.

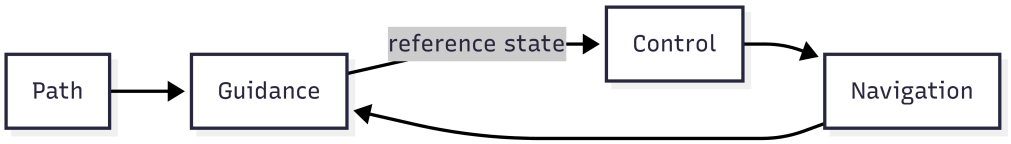


Figure 2.7: Simplified flow diagram for guidance and control

In this project, the focus is more on control, especially for the docking application of an ASV, but there is also a focus on guidance, important for extra safe and accurate positioning in a dynamic marine environment. Guidance can also be used for sensor integration, processing the inputs from the sensors, but that topic is not going to be discussed with further detail due to not being in the range of topics of the dissertation.

2.3.1 Guidance

Trajectory Tracking

Trajectory following control approaches are generally used when time is critical with time-constrained control, as these follow a time-parameterized path, meaning both position and timing are prescribed. The objective is to match both the desired position and velocity (and possibly acceleration) at every instant.

Works with trajectory tracking applications of ASVs are adopting MPC [71] since it allows for the incorporation of future trajectory predictions into the control strategy, enabling a fast response to environmental changes [68]. Methods using MPC sound really promising, especially when compared to traditional methods, for example, the PID which used to be the most used for trajectory following. Malouche & Bouani (2017) conducted a study comparing the performance of constrained MPC with a Multiple Input Multiple Output adaptive PID regulator for trajectory tracking [41], showing that the MPC outperforms the PID in terms of handling constraints and optimizing trajectories in real-time applications.

Path Following and Target Tracking

Everyday path following approaches are being tested in all types of autonomous vehicles, including for ASV docking. In path following methods the controller is never concerned with time as it only tries to follow a spatial path 2.8. The objective with path following is to stay on the path,

regardless of how fast or slow you go, with the main focus usually being to minimize the deviation from the path (lateral or positional error), often quantified as cross-track error [36].

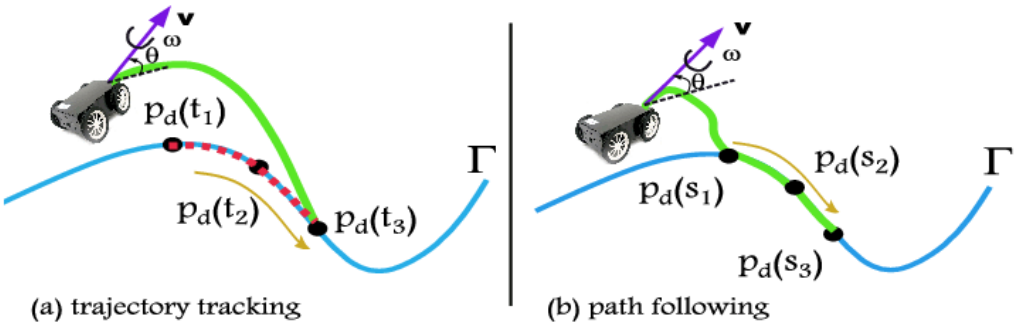


Figure 2.8: Trajectory tracking vs. Path following [49]

Some articles differ path following from target tracking with the difference being that in target following methods, instead of only trying to minimize cross-track error and keeping a constant speed, these try to follow a virtual, ahead of the path, target, effectively guiding the vehicles through smooth turns. Common techniques such as Line of Sight (LOS), Pure Pursuit (PP) or Constant Bearing (CB) belong to this category, with LOS methods, in particular, being known for their robustness, due to being the only ones that take into consideration the distance from the path making them popular in marine vehicle implementations. They define a "lookahead" point along the reference line, and with the control generally aiming towards minimizing the heading to the target but sometimes also aiming to minimize distance to that target instead of heading 2.9. In example, [67] suggest an LOS approach using well-known L1 method for 3D UAV path following while in [17] a an integral LOS approach control for UVs with adaptive compensation for drift forces is suggested.

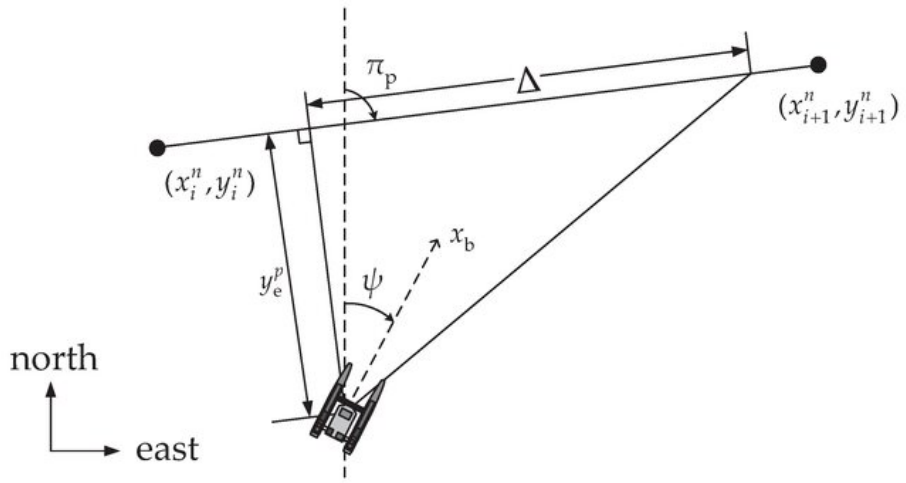


Figure 2.9: Line-of-Sight (LOS) guidance principle. The vehicle at position (x_i^n, y_i^n) steers toward a lookahead point (x_{i+1}^n, y_{i+1}^n) on the desired path. The LOS angle ψ is calculated based on the relative position between the vehicle and the lookahead point, while Δ represents the lookahead distance and π_p denotes the along-track distance. [27]

2.3.2 Control

Control refers to the process of managing the robot's actuators, and so its behavior, in order for it to autonomously achieve the desired objectives such as moving to a target position, following a trajectory, avoiding obstacles, or even maintaining its stability. Control techniques have been used across all different applications for autonomous systems, from controlling the navigation of autonomous vehicles in all types of environments to precise control of industrial manipulators or even in controlling power, voltage, and current in energy systems. For the purpose of this dissertation, the focus will be on control for navigation

There are two main types of control :

- **Open Loop Control**: The controller has no feedback. The robot executes pre-difined commands, assuming everything about the robot and the environment is the same and was known when designing the controller [57].
- **Closed Loop Control**: Operates with feedback 2.10. Sensors are integrated and used to monitor the environment, and the controller dynamically adjusts its actuation based on this information [58] [65].

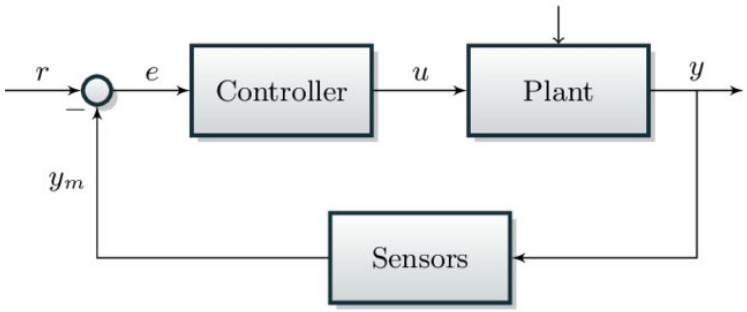


Figure 2.10: Closed Loop Control System Principals [47]

Proportional Integrative and Derivative (PID) control remains one of the simplest and most used techniques in marine applications 2.11. Silva etAl, [51] describe the utility of PID controllers in ASVs for stable motion and control under various conditions. Also, [69] proposes the design of an adaptive PID controller to cope with dynamic changes in the operational environment, such as water currents and wind, improving the performance of the vessel. Interesting approaches like optimized self-adaptive PID speed control for autonomous vehicles to deal with the nonlinearities and disturbances were suggested. Kebbati [30] tested, in 2021, a method that takes advantage of Genetic Algorithms(GA) and Neural Networks(NN) to optimize PID gains, concluding that although GA-PID shows smoother responses, Neural Networks are more robust against disturbances.

Sliding mode control (SMC) is another control technique that has been applied to ASVs [13]. This control technique is usually useful for tasks with precise objectives like trajectory tracking [25] it has issues, for example, the chattering phenomenon which happens when the control signal

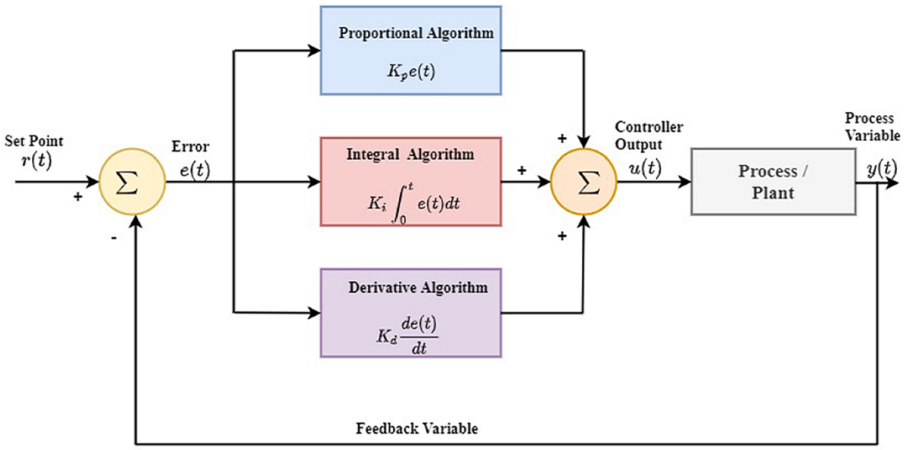


Figure 2.11: PID Control System Diagram [4]

oscillates rapidly around the sliding surface affecting control and stability, and less cost-efficient solutions have to be applied.

In contrast, optimal control techniques like the MPC may not be as well suited for very specific objectives but provides continuous control signals that are less prone to oscillations and is better to deal with more complex tasks, allowing to incorporate diverse constraints without compromising the robustness of the system.

MPC is an advanced control strategy that has gained significant popularity in recent years, especially in fields requiring high performance and adaptive control, such as autonomous vehicles [19, 55], robotic manipulators [10] or even in power systems [6]. MPC operates by predicting the future behavior of a system over a specified time horizon, using a dynamic model of the system to optimize control actions 2.12. Unlike traditional control methods that respond to current states, MPC proactively calculates an optimal sequence of control actions by solving a constrained optimization problem at each control step, that always takes into consideration system restrictions.

.

Artificial Intelligence (AI) or Machine Learning (ML) methods, especially reinforcement learning (RL), deep reinforcement learning (DRL) and neural networks, are also becoming more significant in control systems, including marine vehicles. AI-based controllers work amounts of data to learn optimal control strategies and adapt dynamically to environmental uncertainties or system changes, providing robustness beyond traditional controllers [14, 38]. DRL, in particular, allows autonomous vehicles to learn efficient control policies through trial-and-error interactions with their environments, with methods like Particle Policy Optimization (PPO) gaining more attention over the past few years [31]. These approaches show remarkable performance improvements in complex scenarios with unknown or dynamic conditions.

[5]Image source link

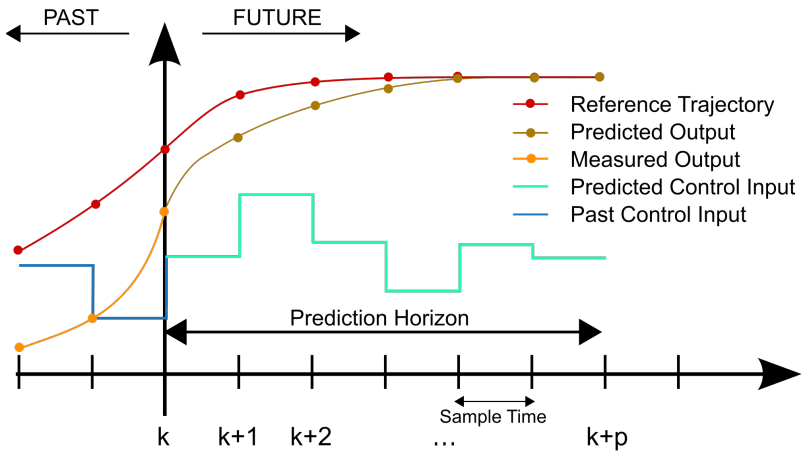


Figure 2.12: Illustration of an MPC prediction and control graph with a reference trajectory. After computing the predicted control sequence (turqoise) only the first input (at k+1) is applied[5]

Hybrid techniques have also gained relevance in research due to the capability of merging strengths of control techniques for specific tasks in order to get complex and very robust control systems. In example, Kebbati [30] tested, in 2021, a method that takes advantage of Genetic Algorithms(GA) and Neural Networks(NN) to optimize PID gains, concluding that although GA-PID shows smoother responses, Neural Networks are more robust against disturbances. Geranmehr & Vafaee [21], suggests an hybrid adaptive neural network controllers incorporating sliding mode techniques to improve robustness against uncertainties in vehicle dynamics, respectively. In [20] a hybrid approach using MPC as a body level controller and SMC as a limb level reference controller is suggested for a multi-legged robot control. On the other hand, Chavez et al. [8] and ML-MPC hybrid approach suggest combining NN with MPC for station keeping control of an ASV achieving faster control execution time and reduced poisition error.

Such hybrid systems show being able to manage complex ASV operations and provide reliable performance in some scenarios

Optimal Control Problem

Solvers

Optimal control methods like the MPC, rely on solving an optimization problem at each time in order to compute the optimal control input every time. In control, the objective of solving such problems is to find a solution in order to try and find the best input for the system.

Typically, the optimization problem is formulated in the form of a [6]**Cost Function** 2.3, where cost matrices define what we want to prioritize.

$$J = \sum_{t=0}^{N} \left(x_t^\top Q x_t + u_t^\top R u_t \right) \tag{2.3}$$

[6]These are also commonly called **Objective Functions**

In the given example, it can be seen a quadratic cost function formulation, where Q and R are the cost matrices and these are usually chosen keeping in mind certain performance criteria like minimizing deviations from a target state or reducing the energy consumption of the control inputs. A larger Q would emphasize tighter state regulation, while a larger R would penalize excessive or aggressive control actions.

Note that, a cost function might not be quadratic, sometimes even being highly nonlinear, and to solve those optimization problems, there are diverse approaches and solutions, all of them already extensively tested which pros and cons were already discussed for each different problem requirement :

- Quadratic Problem Solvers: Introduced in 1956 by Frank and Wolf [18], these methods became widely used for optimization problems whose cost function is quadratic and constraints are linear 2.3. QP solvers are crucial to solve Linear Quadratic Regulator (LQR) problems, commonly used in control systems for trajectory tracking optimization.

- Nonlinear Programming (NLP) Solvers: For systems with nonlinear dynamics or non-quadratic cost functions 2.4, the optimization becomes nonlinear. The (Interior Point Optimization) IPOPT solver is the most commonly used in this context [60].

$$\begin{aligned} \min_{x} \quad & J(x) = \sum_{t=0}^{N} \ell(x_t, u_t) \\ \text{subject to:} \quad & g(x) = 0, \\ & h(x) \leq 0 \end{aligned} \tag{2.4}$$

- Nature-inspired solvers: There are different types of nature-inspired optimal control solvers.

  A general for such optimization problem is 2.6.

$$J = \sum_{t=0}^{N} \ell(x_t, u_t) \tag{2.5}$$

$$\min_{x} \quad J(x) \text{ subject to:} \qquad x \in S \tag{2.6}$$

  where S is the feasible search space and *J(x)* 2.5 is the total cost. These have the advantage of being gradient-free methods, and, contrarily to the other gradient-based methods, the minimum x is found by testing all the feasible possibilities and understanding which results in a lower final cost.

  Particle Swarm Optimization (PSO) is a social-behavior inspired optimization technique, where articles cooperate and share information about good solutions through social learning 2.13. It is very good for nonlinear and non-convex optimization problems, see [61], but it may be computationally very expensive and not output the optimal solution "due to the limitation of the search space of particles" [9].

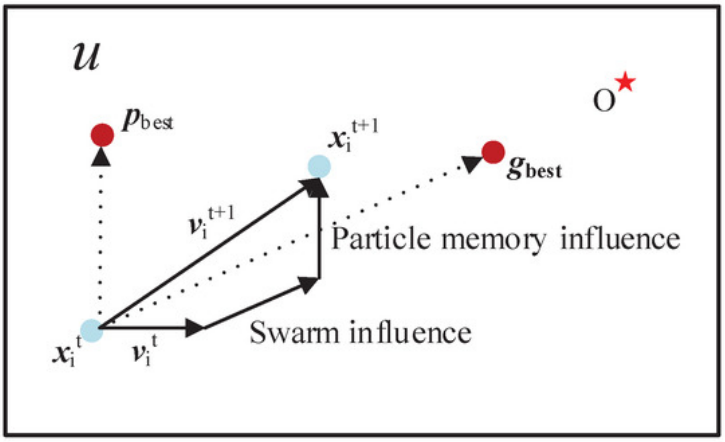


Figure 2.13: Particle Swarm Optimization (PSO) algorithm dynamics for position planning. Each particle moves from position x_i^t to x_i^{t+1} with velocity v_i^{t+1} influenced by its personal best solution (p_{best}) and the swarm's global best solution (g_{best}). The dotted lines represent particle memory influence and swarm influence components. [66]

Another one of this methods is Genetic Algorithm (GA). GAs where the first nature-inspired methodology and can also be very useful for solving highly nonlinear and complex optimization problems with multiple objectives. These differ fro PSO in the sense that, individual particles do not 'communicate', and instead, compete for survival, with only the fittest passing genes to next generation In the optimization context, context GAs can be used to optimize trajectories or control inputs in systems, where gradient-based methods fail. It has the problem of being computationally expensive and seemingly not suitable for real-time applications such as MPC but with some fine tuning for the specific objective, it's possible to overcome this problem. Integration of GAs in conjunction with other optimization techniques have also been suggested [33].

2.4 Related Work

Over the years, solutions to guidance and control systems for navigation and docking have been proposed using various methods. Early approaches to ASV docking considered fuzzy control systems [56], although these required significant computational resources and lacked extensive testing across diverse docking scenarios, despite showing promising initial results. In contrast, Lexau et al. [37] suggests using PID with LOS guidance as a GNC approach for docking large ships under wind and current disturbances.

In recent years optimal-control solutions became the most mentioned topic for navigation and docking in marine environments, though, nowadays AI methodologies are also being strongly researched and applied in the field.

Classical path following approaches include Line-of-Sight (LOS) guidance methods, which represent one of the most fundamental approaches in marine vehicle guidance. Nilsson et al. [44] suggests using geometric relationships to create a Voronoi diagram with guidance waypoints that

generate heading commands and taking advantage of MPC frameworks to control the vehicle. A wide variety of MPC-based approaches have been developed for ASV trajectory tracking and docking applications, for example, Zheng et al. [70] present both nonlinear MPC (NMPC) and linearized MPC (LMPC) formulations for two-dimensional trajectory tracking of autonomous marine surface vehicles, demonstrating the effectiveness of MPC in handling constraints and optimizing trajectories in real-time applications. Their work provides a comprehensive comparison between nonlinear and linearized approaches, showing computational complexity trade-offs. Martinsen et al. [42] propose a method for performing autonomous docking using numerical optimal control, framing the task as a dynamic positioning problem with spatial constraints for collision avoidance. Their approach presents a complete method for docking, maneuvering, dynamic positioning, and control allocation, implemented as a real-time MPC-based algorithm, in a very concise approach. In the article [43], researchers also mention an MPC approach for docking, integrating sensors, and although the sensors are not in the range of this dissertation, the mentioned guidance and control techniques were also very useful for the development of this work.

Recent advances in guidance systems for ASV docking include the work by Sun et al. [53], who developed a novel SARLOS guidance system for berthing control with collision avoidance, which is a twist in the already suggested . Their approach incorporates control barrier functions (CBF) to transform collision avoidance with docks into constraints on ship speed and yaw rate, establishing (QP) formulations to calculate optimized reference signals.

For environments with multiple obstacles, Ødven et al. [45] present hybrid methods that address docking while avoiding both static and dynamic obstacles. Their approach utilizes computational geometry for partitioning non-convex harbor areas into convex polygons, followed by search algorithms and numerical optimal control for trajectory planning.

Optimization-based motion planning for marine vessels in complex environments has also been explored in [2] by Bergman et al., who propose algorithms for automatic computation of spatial safety envelopes around computed trajectories. Their work addresses the transition from optimization-based motion planning to optimization-based control for docking operations.

The recent advancements have also been getting attention, with researchers applying these techniques in autonomous docking. Løver et al. [40] implemented explainable AI methods for deep reinforcement learning agents performing automatic docking. Their work addressed the black-box nature of artificial neural networks by implementing Kernel SHAP, LIME, and Linear Model Trees to provide explanations for decisions made by the docking agent trained via Proximal Policy Optimization.

Visual guidance is also a very discussed topic, with techniques rely on visual feedback for high precision docking. Page et al. [46] noted that visual-based methods generally provide the best accuracy near docking stations and are therefore suitable for situations where precise localization is required. Yet environmental factors such as lighting and visibility (which are very common in marine applications) can affect the effectiveness of visual data and limit its application in some situations.

2.5 Summary and Analysis

Since the MPC may already need some computational power the RRT would possibly bring the most advantage since the its intended to work on a complex and dynamic environment and even though it may not output the shortest path, but docking problem does not deal with long paths, so small differences, if they happen, should not be a significant problem.

In this chapter, we exposed and reviewed various guidance and control techniques for ASV docking. Firstly, it was considered the relative strengths and limitations of classical planning methods. Graph-based methods like Dijkstra's and A*, or even more advanced methods like the D* guarantee optimality in static environments but struggle with real-time changes and had the requirement of map discretization which creates an extra layer of complexity. Sampling-based approaches like RRT or PRM on the other hand, don need map discretization but often deliver suboptimal trajectories although being fast. Then we introduced RRT*, a twist on the classic RRT algorithm that improves the quality of the generated trajctories. Then, potential-field planners were also discussed, appearing as a strong methodology bringing real-time adaptability and smoother paths, but with the handicap of being possibly stuck in local minima, though the problem already being partially tackled with the introduction of Navigation Functions.

Then the documentation turned to hybrid schemes that combine real-time trajectory optimization with global path generation, and focusing on how the chosen combination addressed computational complexity, environmental disturbances to result in smooth, collision-free docking maneuvers. Guidance algorithms were discussed, with highlight for LOS algorithms, where a virtual look-ahead point is selected on the reference path and the vessel's heading is continuously controlled to minimize point-to-point error, wether by heading or position control. Afterwards, relevant theory was exposed, clearly mentioning the difference between path following, where progress is tracked, and timing-required trajectory following.

Moving beyond pure path-planning, this chapter dove into classical control techniques (PID and sliding-mode control) that translate these reference paths into actuator commands: PID is the most well-known control method for its simplicity and proven stability despite tuning challenges. Then it was also mentioned SMC which is robust to disturbances but has the cost of possible chattering.

Finally, optimization-based controllers were introduced, giving more relevance to Model Predictive Control (MPC) for its powerful handling of multi-variable constraints over a receding horizon. In the context of this topic, it was also mentioned optimal control problem (OCP) solvers, with highlight to nonlinear solvers, that allow the integration of more complex and realistic systems, but still with the handicap of possibly falling in local minima solutions and nature-based solvers, which may solve the local minima solution problem, although harder and more expensive to compute.

Emerging AI/DRL techniques were also mentioned, as these are able to learn control policies directly from data and manage how to balance computational load, disturbance rejection, and control stability, with other possible capabilities yet to be found, but regularly being studied.

The trend toward hybrid approaches reflects the recognition that no single methodology can address all challenges inherent in autonomous marine operations. The integration of path planning with control systems, environmental disturbance compensation, and real-time constraint handling has become essential for the practical deployment of autonomous docking systems.

Chapter 3

Homing and Docking Control Framework for ASVs

The autonomous motion of surface vehicles in maritime environments presents significant challenges due to the complex nonlinear dynamics, environmental disturbances, and safety-critical constraints inherent to marine systems. Docking is transversal to all missions where maritime vessels are used, being not only one of the most important tasks of the mission but also one of the most meticulous ones due to it's proximity to obstacles.

MPC has emerged as a paradigmatic approach for handling such multi-objective control problems with constraints. However, the computational burden associated with NMPC often precludes real-time implementation, particularly for systems requiring high sampling frequencies.

This paper proposes a sequential linearization MPC (SL-MPC) framework that achieves an efficient compromise between control performance and computational efficiency. The methodology systematically linearizes the nonlinear ASV dynamics at each control step, enabling the application of a quadratic NLP solver while maintaining adequate fidelity to the underlying nonlinear system behavior.

This work also understands the limited amount of test possibilities for ASV motion using only the controller given an initial and a desired state, specially in constrained harbor scenarios so, a complete GNC pipeline was developed as well as Path Planning and Pre-Planning methodologies, introducing the homing process (moving for an area close to the desired docking position).

This chapter starts by explaining the complete system architecture, exposing how each subsystem interconnects. It then proceeds to dive deep into the developed solution for each layer starting from map decomposition, path planning and map interaction, and ending with the core of the work, the GNC algorithms achieving a complete functional docking strategy for ASVs.

3.1 System Architecture

3.1.1 Hierarchical System Framework

The autonomous docking system employs a hierarchical control architecture that decomposes the complex navigation and docking problem into manageable subsystems operating at different temporal and spatial scales. This decomposition follows established principles in autonomous vehicle control, where high-level strategic decisions are made at slower rates while low-level reactive control operates at higher frequencies to ensure system responsiveness and safety. The system architecture proposed in this work is defined 3 main hierarchical subsystems. Figure 3.1 illustrates a simplified version of the complete system workflow, showing how these subsystems connect.

Planning Layer: This layer generates collision-free reference trajectories from the vessel's current position to the desired docking location. This layer processes harbor map, deconstructing it into static obstacle constraints, which are then used to produce a sequence of $[x, y]$ coordinates that connect the initial and last positions through a collision-free path.

- **Environmental Mapping Module:** This module takes a bird's eye view image of the harbor as an input, and processes it, making a realistic binary occupancy map, defining obstacles (as polygons) and motion allowed areas. It uses Douglas-Peucker Algorithm for vertex reduction and afterwards it is performed obstacle dilation for safety, to ensure conservative motion while accounting for vehicle dimensions and positioning uncertainties.
- **Artificial Potential Field Module:** This module repulsive forces that prevent the vehicle from hitting static obstacles. It uses distance from polygons to calculate the repulsive force, which is then used to generate a cost function that penalizes proximity to obstacles.
- **Global Path Planning Module:** This module generates globally optimal or near-optimal collision-free paths (sequence of nodes) from arbitrary starting positions to specified docking locations.
- **Path Smoothing Module:** This module uses the predefined nodes and creates a continuous path that passes through all of them using interpolation curves.

GNC Layer: This layer performs real-time trajectory optimization and control command generation. The guidance module generates a reference state automatically sent to the MPC algorithm, which continuously solves the finite-horizon optimal control problem to track the reference while adapting to dynamic conditions and disturbances.

- **Guidance Module:** Sometimes considered as part of local planning layer, it acts as the interface between global planning and control. It computes the reference which is then sent to the controller.
- **Model Predictive Control Module:** This module is the heart of the system, as it defines in real-time the best action to take in every different dynamic situation. This module implements the SL-MPC which performs performs iterative optimization of control sequences,

and generates thruster commands that minimize tracking errors while respecting physical constraints.

Motion Layer: This layer represents the pipeline after the output from the control system. This consists in applying the optimal thruster's force, given by the MPC and then apply it to the vehicle in order to induce motion and a consequent new state. in a real-world scenario the ASV would move and the new state estimations would be calculated and used to feed Guidance and to serve as new operating point. Since this work focuses on simulation, the real-world ASV motion is simulated through a realistic function of the vehicle dynamics which, when fed with the control input, outputs the new state.

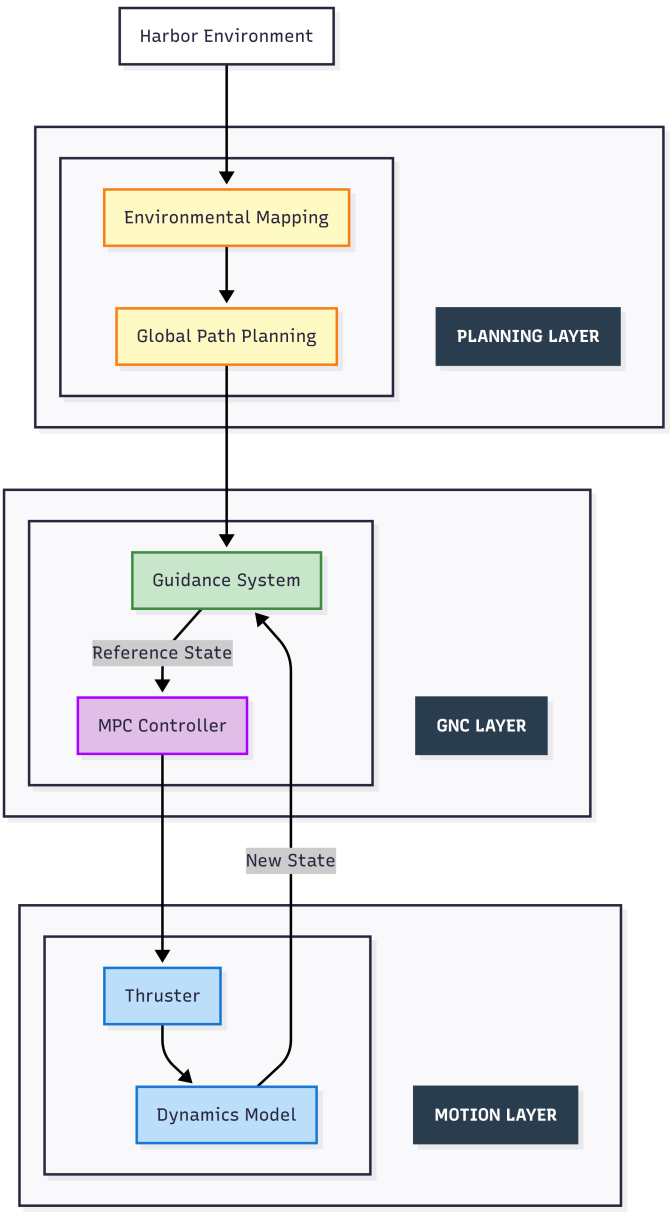


Figure 3.1: System Workflow Diagram

For a better understanding of each layer, from this moment on, this dissertation will follow the color scheme presented in 3.1 for each Module and subsystem.

- Planning Layer
- Guidance Layer
- MPC Controller Layer
- Motion Layer

3.1.2 Real-Time GNC Loop Architecture

The GNC Layer implements the following sequence at each control iteration:

Reference State Generation: The guidance algorithm provides reference state $\mathbf{x}_d$ on the planned trajectory based on the current position of the vehicle. Reference generation includes look-ahead logic to ensure smooth trajectory following and appropriate approach speeds for docking maneuvers.

Sequential Linearization: The nonlinear ASV dynamics are linearized around the current operating point (x_0, u_0), generating time-varying linear models suitable for convex optimization. This linearization process captures the essential nonlinear characteristics while maintaining computational tractability.

Discretization: The continuous-time linearized dynamics must be converted to discrete-time form suitable for digital implementation. It allows the MPC algorithm to compute control actions at discrete sampling instants while maintaining adequate approximation accuracy for the ASV's dynamics.

Optimization Problem Formulation: The MPC optimization problem is constructed using the linearized dynamics, current state measurements, and reference signals. The optimization variables include the control sequence over the control horizon, while the cost function balances trajectory tracking, control effort, and constraint satisfaction.

Optimal Control Computation: The optimization problem is solved using interior point methods implemented through CasADi's automatic differentiation framework. This step is where it is possible to add and tune parameters and constraints in order to achieve a good real-time performance while maintaining convergence guarantees.

Control Implementation: The first element of the optimal control sequence is applied to the vehicle's thruster system, while the remaining sequence provides warm-start initialization for the subsequent optimization iteration.

Figure 3.2 shows the iterative loop that forms the heart of the real-time control system. The image shows the closed-loop interaction between the MPC controller, the ASV Motion, and the guidance system, highlighting the GNC Layer, which is the main focus of this work.

Symbolic Model Representation using CasADI: The control system implementation leverages CasADi for symbolic computation and automatic differentiation, enabling efficient gradient-based optimization while maintaining code clarity and numerical accuracy. Vehicle dynamics, cost

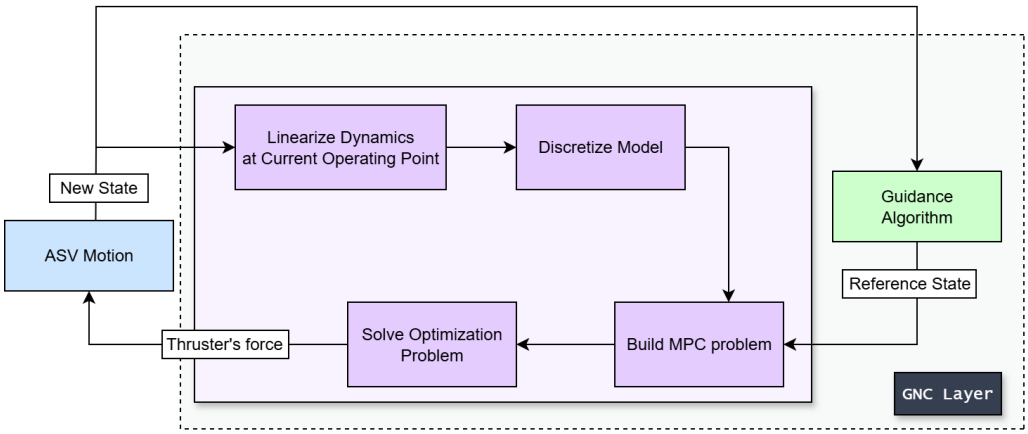


Figure 3.2: Detailed diagram of the MPC Module with highlight of the GNC subsystem

functions, and constraints are represented symbolically, allowing CasADi to automatically generate optimized code for function evaluation and derivative computation. This approach eliminates hand-coded gradients while ensuring numerical accuracy and computational efficiency.

3.1.3 Performance and Safety Considerations

The system architecture incorporates layers of safety mechanisms and performance monitoring to ensure reliable operation in the challenging marine environment:

Constraint Handling: Physical limitations of the thruster system, maximum approach speeds are enforced through hard constraints in the MPC formulation. Minimum safe distances from obstacles can also be enforced through hard constraints, although that could make the optimization problem too complex and not real-time feasible.

Real-Time Performance Monitoring: Computational timing, convergence metrics, and control performance indicators are continuously monitored to detect potential system malfunction.

Figure 3.3 shows how the GNC and Motion layers interact for the simulation and therefore, system monitoring.

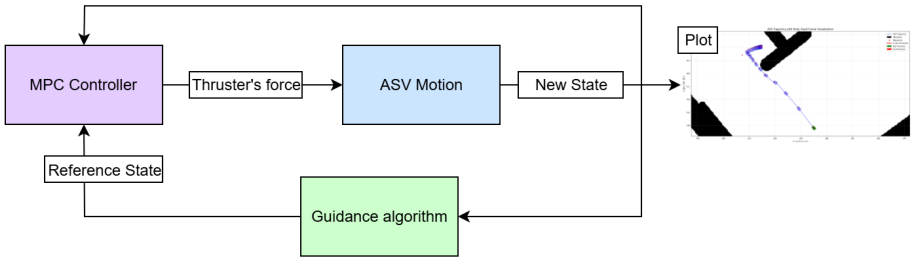


Figure 3.3: Guidance, Control, Motion and Plot relations

The main focus of the dissertation lies in the guidance and control system. For simulation, the nonlinear version of the system dynamics is executed with the optimal control obtained from

the MPC controller and then, the new simulated state is used for both motion visualization, monitorization, and input for the guidance module in order to obtain the closed loop GNC system, required for the real world system.

3.2 Introduction to MPC

1. **Prediction:** MPC relies on a mathematical model of the ASV's dynamics to predict its future states. In the context of docking, it allows MPC to anticipate how the ASV will respond to control actions, enabling smoother and safer maneuvers.

2. **Optimization :** At each control step, an optimization problem is solved to find, for each thruster, the control inputs that minimize a desired cost function while satisfying system constraints. This solution is evaluated over a short period, known as the **prediction horizon**, to anticipate future states. However, it is only implemented for an even shorter period, called the **control horizon**, before being recalculated based on updated sensor data. This approach allows us to save unnecessary computational power and enables real-time adaptability to dynamic conditions.

3. **Actuation:** After the optimization process determines the optimal control inputs, the first control input from the optimized sequence is applied to the ASV. This step is crucial as it guides the vessel's movement toward the docking target, ensuring that the actions taken align with the predictions made during the optimization phase.

4. **Update:** Following the execution of the control input, the ASV updates its state estimates based on the simulated nonlinear dynamics equations. The MPC loop then repeats, recalculating a new thruster input command, thereby continuously adapting to any changes in the environment or the ASV's state.

Figure 3.4 shows a simplified scheme of an general MPC framework . Reference signal guides the optimizer, which uses a vehicle dynamics model to predict the optimal control actions. These actions are applied to the plant (ASV), adjusting variables like speed and position based on feedback.

The MPC controller represents the core of this system, as every upper layer calculations for reference signals are done in order to allow a better performance of the controller, thus, better overall system performance.

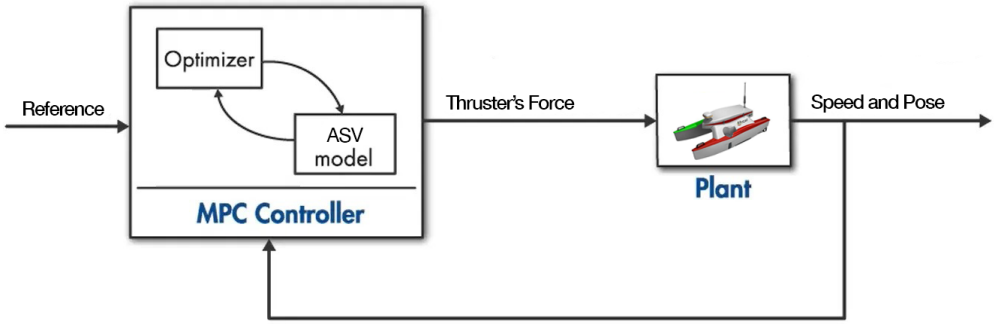


Figure 3.4: MPC generalized scheme

3.3 Mathematical Modeling

3.3.1 Vehicle Dynamics

In modeling an autonomous surface vehicle (ASV) operating in the horizontal plane, the general approach goes for a balance between capturing the essential physics of 6-DOF motion and reducing the complexity by neglecting z-axis motion, heave (vertical translation), roll (rotation about the longitudinal axis), and pitch (rotation about the transverse axis). This reduction yields a 3-DOF model in surge (forward), sway (sideways), and yaw (rotation about the vertical axis). Below, there's a deeper explanation of each component of the model, providing both mathematical form and physical insight, and setting the stage for model predictive control (MPC) design.

Coordinate Frames and State Definition

For our ASV moving in the horizontal plane, we adopt a 6-DOF body-fixed model(reduced to planar motion).

Then we defined our full state 3.1, which is the typical formulation for these problems, taking only into account $[x \ y \ \psi]$ and the speed information $[u \ v \ r]$, since the **z**-axis information is not necessary for planar control. Then we obtain the full-state formulation for the problem, as shown in eqaution 3.3.

$$\mathbf{x} = \begin{bmatrix} \eta \\ \nu \end{bmatrix} \tag{3.1}$$

where η and ν are :

$$\eta = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}, \qquad \nu = \begin{bmatrix} u \\ v \\ r \end{bmatrix}. \tag{3.2}$$

Thus, the 6-dimensional full-state is:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \psi \\ u \\ v \\ r \end{bmatrix} \tag{3.3}$$

- (x, y) are global NED coordinates (m).
- ψ is the yaw angle (rad), measured from true north.
- (u, v) are surge and sway velocities (m/s) in the body frame.
- r is the yaw rate (rad/s), the time derivative of ψ.

Kinematic Relations

As mentioned, in equations (3.1, 3.2, 3.3), (x, y) are earth-fixed positions, ψ is heading, and (u, v, r) are surge, sway, and yaw-rate in the body frame. To understand how the vehicle behaves in the earth frame we have to map how body-fixed frame relates with earth-fixed frame. This relationship can be seen in Figure 3.5.

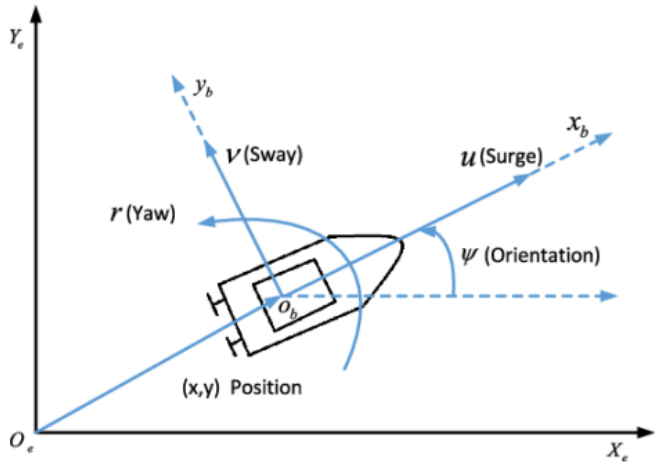


Figure 3.5: Relation between Earth fixed frame (East-North-Down ENU) system with vehicle Body fixed frame

Kinematics describe how body-frame velocities translate into motion in the inertial frame. This mapping is accomplished by a rotation matrix $\mathbf{J}(\psi)$.

$$\mathbf{J}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.4}$$

Knowing the rotation matrix we can then define the relation between η and ν represented the following equation 3.5

$$\dot{\eta} = \mathbf{J}(\psi)\nu, \tag{3.5}$$

This rotation highlights that surge and sway velocities projected into the ENU frame depend directly on the vessel's heading. The yaw rate component completes the kinematic description: $\dot{\psi} = r$.

Dynamics Equations

The dynamic behavior of the ASV in the body-fixed frame is governed by the Newton-Euler equations, which can be expressed in the compact form:

$$(M_{RB} + M_A)\dot{\nu} + (C_{RB}(\nu) + C_A(\nu) + D(\nu))\nu = \tau \tag{3.6}$$

This fundamental equation represents the balance of forces and moments acting on the vessel, where each term captures distinct physical phenomena that must be carefully modeled for accurate control design. The total system inertia M, total Coriolis matrix $C(\nu)$, and the generalized force vector τ are given by:

$$M = M_{RB} + M_A \tag{3.7}$$

$$C(\nu) = C_{RB}(\nu) + C_A(\nu) \tag{3.8}$$

$$\tau = T \times u \tag{3.9}$$

where T is the control allocation matrix and u represents the thruster commands.

Inertia Force Components

Rigid-Body Inertia (M_{RB})

The rigid-body inertia matrix represents the vessel's inherent resistance to acceleration due to its mass distribution and rotational inertia. In the general case, this matrix accounts for the vessel's mass m, center of gravity location z_G, and moment of inertia I_z:

$$M_{RB} = \begin{bmatrix} m & 0 & mz_G \\ 0 & m & 0 \\ mz_G & 0 & I_z \end{bmatrix} \tag{3.10}$$

For the ASV under consideration, with the simplifying assumption that the center of gravity coincides with the origin of the body-fixed coordinate system ($z_G = 0$), this reduces to:

$$M_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{3.11}$$

where:

- m (kg) represents the total vessel mass
- I_z (kg) $\cdot$m^2 is the moment of inertia about the vertical (yaw) axis

The units indicate: surge and sway inertia in kg; yaw inertia in kg$\cdot$m^2.

Added Mass Effect (M_A)

Added mass represents the additional inertia experienced due to the acceleration of surrounding fluid that moves with the vessel. This phenomenon is particularly significant for marine vehicles operating at the air-water interface, where the vessel must accelerate not only its own mass but also a volume of water that moves coherently with the hull.

The added mass matrix in its general form is:

$$M_A = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -N_{\dot{v}} & -N_{\dot{r}} \end{bmatrix} \tag{3.12}$$

The units represent: (1,1) surge added mass in kg; (2,2) sway added mass in kg; (3,3) yaw added inertia in kg$\cdot$m^2; (2,3) and (3,2) sway-yaw coupling in kg$\cdot$m.

The physical interpretation of each coefficient is:

- $X_{\dot{u}}$: Added mass in surge direction due to surge acceleration, representing the water mass accelerated forward/backward with the vessel
- $Y_{\dot{v}}$: Added mass in sway direction due to sway acceleration, significantly larger than surge due to the bluff body characteristics in lateral motion
- $N_{\dot{r}}$: Added rotational inertia due to yaw acceleration, accounting for the rotational acceleration of fluid around the vessel
- $Y_{\dot{r}} = N_{\dot{v}}$: Cross-coupling terms between sway velocity and yaw rate, representing the dynamic interdependence of lateral and rotational motions

The **total system inertia matrix** becomes:

$$M = M_{RB} + M_A = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix} \text{ kg, kg}\cdot\text{m, kg}\cdot\text{m}^2 \tag{3.13}$$

Coriolis and Centripetal Force

Rigid-Body Coriolis Matrix ($C_{RB}(\nu)$)

This matrix captures the non-inertial nature of the body-fixed reference frame. It represents the Coriolis and centrifugal forces which come from the vessel's motion in the rotating body-fixed coordinate frame.

In general form:

$$C_{RB}(\nu) = \begin{bmatrix} 0 & 0 & -m(z_G r + v) \\ 0 & 0 & mu \\ m(z_G r + v) & -mu & 0 \end{bmatrix} \tag{3.14}$$

For the simplified case where the center of gravity is at the coordinate origin ($z_G = 0$):

$$C_{RB}(\nu) = \begin{bmatrix} 0 & 0 & -mv \\ 0 & 0 & mu \\ mv & -mu & 0 \end{bmatrix} \text{ N} \tag{3.15}$$

These terms represent:

- $-mv$: Centrifugal force in surge due to sway velocity
- mu: Centrifugal force in sway due to surge velocity
- **Cross terms**: Coupling between translational and rotational motions

Hydrodynamic Coriolis Effect ($C_A(\nu)$)

This matrix represents the Coriolis forces that come from from the added mass effects, essentially the "centrifugal forces" experienced by the accelerated fluid mass:

$$C_A(\nu) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}} v + Y_{\dot{r}} r \\ 0 & 0 & -X_{\dot{u}} u \\ -Y_{\dot{v}} v - Y_{\dot{r}} r & X_{\dot{u}} u & 0 \end{bmatrix} \text{ N} \tag{3.16}$$

The total Coriolis matrix is then:

$$C(\nu) = C_{RB}(\nu) + C_A(\nu) = \begin{bmatrix} 0 & 0 & -(m - Y_{\dot{v}})v - Y_{\dot{r}} r \\ 0 & 0 & (m - X_{\dot{u}})u \\ (m - Y_{\dot{v}})v + Y_{\dot{r}} r & -(m - X_{\dot{u}})u & 0 \end{bmatrix} \text{ N} \tag{3.17}$$

Hydrodynamic Damping Matrix

The damping matrix $D(\nu)$ represents the velocity-dependent hydrodynamic forces that oppose the vessel's motion. For marine vehicles, these forces are predominantly quadratic in velocity, reflecting the turbulent nature of flow separation at moderate to high Reynolds numbers.

The damping matrix takes the form:

$$D(\nu) = \begin{bmatrix} -X_{u|u|}|u| & 0 & 0 \\ 0 & -Y_{v|v|}|v| & -Y_{r|r|}|r| \\ 0 & -N_{v|v|}|v| & -N_{r|r|}|r| \end{bmatrix} \tag{3.18}$$

The physical significance of each coefficient:

- X_{uu}: Surge drag coefficient, relatively small due to the streamlined hull design in the forward direction
- Y_{vv}: Sway drag coefficient, significantly larger than surge drag due to bluff body effects when moving sideways
- N_{rr}: Yaw damping coefficient, providing rotational stability
- $Y_{rr} = N_{vv}$: Cross-coupling terms between sway velocity and yaw rate, representing the complex hydrodynamic interactions during turning maneuvers

Thruster Allocation

The thrust allocation matrix T establishes the relationship between individual thruster commands and the generalized forces and moments applied to the vessel:

$$\tau = T \times u \tag{3.19}$$

where:

$$\begin{bmatrix} X_{force} \\ Y_{force} \\ N_{moment} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & -1 \\ r_1 & -r_1 & -r_3 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_{tunnel} \end{bmatrix} \tag{3.20}$$

This thrust allocation provides:

- **Surge force**: Algebraic sum of both bow thrusters ($T_1 + T_2$), providing redundancy and higher maximum thrust capability
- **Sway force**: Generated exclusively by the tunnel thruster ($-T_{tunnel}$), making lateral positioning more challenging
- **Yaw moment**: Created by differential bow thruster operation ($0.335(T_1 - T_2)$) combined with tunnel thruster contribution ($-0.3T_{tunnel}$)

This can then, be compactly expressed as:

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu = Tu \tag{3.21}$$

Combined with the kinematic relationship, the complete 6-DOF ASV model is:

$$\dot{\eta} = J(\psi)\nu \tag{3.22}$$

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu = Tu \tag{3.23}$$

This comprehensive dynamic model provides the nonlinear foundation for the sequential linearization approach employed in the MPC implementation, where these complex dynamics are linearized around successive operating points to enable real-time optimization while preserving essential model fidelity for trajectory control.

The **complete dynamics model**, with the identified parameters substituted into each matrix component, can be seen in Appendix A.

3.3.2 MPC Controller

State-Space Model

The state-space representation provides a unified framework for modeling both linear and nonlinear dynamical systems via first-order differential equations.

In its most general continuous-time form, we write:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{3.24}$$

$$y(t) = Cx(t) + Du(t), \tag{3.25}$$

where:

- $x(t) \in \mathbb{R}^n$ is the *state* vector capturing the internal dynamic variables at time t,
- $u(t) \in \mathbb{R}^m$ is the *input* (control) vector of external forces or commands,
- $y(t) \in \mathbb{R}^p$ is the *output* vector, typically a measurable subset or transformation of the state,
- $A \in \mathbb{R}^{n\times n}$ is the *state matrix*, defining autonomous dynamics,
- $B \in \mathbb{R}^{n\times m}$ is the *input matrix*, coupling controls into the state,
- $C \in \mathbb{R}^{p\times n}$ is the *output matrix*, projecting state to outputs,
- $D \in \mathbb{R}^{p\times m}$ is the *feedthrough* or *direct-transmission matrix*.

Equations 3.24–3.25 form the foundation for modern control design, enabling tools such as pole placement, observer design, and optimal control formulations While this linear framework forms the theoretical foundation for much of modern control theory, real-world systems like marine vehicles exhibit inherently nonlinear behavior that cannot be adequately captured by linear models. The challenge lies in extending these concepts to handle nonlinear dynamics while maintaining computational tractability for real-time applications.

Nonlinear System Dynamics and ASV Modeling

Transitioning from linear to nonlinear systems represents a fundamental shift in both mathematical complexity and computational requirements. For autonomous surface vehicles, the nonlinear nature of the dynamics come from a variety of reasons: coupling between translational and rotational motions, the velocity-dependent nature of hydrodynamic forces, and the often complex interactions between the vehicle and the marine environment.

The general nonlinear state-space representation extends the linear framework by replacing the linear functions with nonlinear mappings:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{3.26}$$

where $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a nonlinear vector function that encapsulates the complete system dynamics. Unlike the linear case, where system behavior is completely characterized by constant matrices, the nonlinear function $\mathbf{f}$ can exhibit complex dependencies on both state and input variables.

For our autonomous surface vehicle application, the state vector combines position, orientation, and velocity information 3.3.

The nonlinear function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ encapsulates the complete physics of marine vehicle motion. As mentioned in the previous chapter, this includes the kinematic relationships between body-fixed velocities and earth-fixed position changes.

Sequential Linearization

The sequential linearization approach represents a crucial method for managing the computational complexity of nonlinear MPC while preserving essential system characteristics. The importance of this approach is that, while the system may exhibit strong nonlinearities over large regions of the state space, it can be well-approximated by linear models over the smaller regions traversed during a single control iteration.

For linearization, the mathematical equation rests on the Taylor series expansion of the nonlinear function around a chosen operating point. For a general nonlinear function $\mathbf{f}(\mathbf{x}, \mathbf{u})$, the Taylor series expansion around the point $(\mathbf{x}_k, \mathbf{u}_k^{(0)})$ yields:

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k^{(0)}) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\mathbf{x}_k, \mathbf{u}_k^{(0)})} (\mathbf{x} - \mathbf{x}_k) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{(\mathbf{x}_k, \mathbf{u}_k^{(0)})} (\mathbf{u} - \mathbf{u}_k^{(0)}) + \mathscr{O}(\|\mathbf{x} - \mathbf{x}_k\|^2, \|\mathbf{u} - \mathbf{u}_k^{(0)}\|^2) \tag{3.27}$$

The first-order approximation, obtained by neglecting higher-order terms, provides the linearized model:

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) \approx \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k^{(0)}) + \mathbf{A}_k(\mathbf{x} - \mathbf{x}_k) + \mathbf{B}_k(\mathbf{u} - \mathbf{u}_k^{(0)}) \tag{3.28}$$

The Jacobian matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ are fundamental to the linearization process:

$$\mathbf{A}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\mathbf{x}_k, \mathbf{u}_k^{(0)})} \in \mathbb{R}^{6\times 6} \tag{3.29}$$

$$\mathbf{B}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{(\mathbf{x}_k, \mathbf{u}_k^{(0)})} \in \mathbb{R}^{6\times 3} \tag{3.30}$$

The state Jacobian $\mathbf{A}_k$ characterizes how perturbations in the state variables affect the rate of change of the system. On the other hand, the input Jacobian $\mathbf{B}_k$ describes the influence of control inputs in the system behavior.

For the ASV configuration, this matrix reflects the control allocation from individual thruster forces to generalized forces and moments, modified by the current operating conditions. The sequential nature of the linearization approach addresses the fundamental limitation of static linearization, which is the degradation of approximation quality as the system evolves away from the linearization point. To solve this problem, the system is re-linearized at each sampling instant around the current state in order to maintain approximation accuracy while enabling efficient optimization.

Discretization

The transition from continuous-time system models to discrete-time representations suitable for digital implementation requires careful consideration of numerical integration methods. The choice of discretization method impacts both accuracy and computational efficiency, making it a key design decision in MPC implementation.

The fundamental challenge in discretization lies in accurately representing the continuous-time dynamics through a discrete-time model that can be efficiently evaluated in numerical optimization. For linear systems, exact discretization methods based on matrix exponentials provide theoretically perfect representations. However, for nonlinear systems or time-varying linear systems (as arise in sequential linearization), approximate methods must be employed.

The zero-order hold (ZOH) assumption represents the most common approach for discretizing control systems. This assumption models the control input as remaining constant over each sampling interval, which accurately reflects the behavior of digital control systems where control commands are updated at discrete time instants.

Under the ZOH assumption, the discrete-time equivalent of the linearized continuous-time system can be derived through integration. For the linearized system $\dot{\mathbf{x}} = \mathbf{A}_k\mathbf{x} + \mathbf{B}_k\mathbf{u} + \mathbf{f}_0$, the exact discrete-time representation is:

$$\mathbf{x}_{k+1} = e^{\mathbf{A}_k T_s}\mathbf{x}_k + \left(\int_0^{T_s} e^{\mathbf{A}_k \tau} d\tau\right)\mathbf{B}_k\mathbf{u}_k + \left(\int_0^{T_s} e^{\mathbf{A}_k \tau} d\tau\right)\mathbf{f}_0 \tag{3.31}$$

While this exact representation is theoretically appealing, the computation of matrix exponentials can be expensive, particularly when performed repeatedly in real-time applications. For the sampling rates typical in marine vehicle control (10-100 Hz), first-order approximations provide adequate accuracy with significantly reduced computational cost.

The first-order (Euler) approximation replaces the matrix exponential with its first-order Taylor expansion:

$$e^{\mathbf{A}_k T_s} \approx \mathbf{I} + \mathbf{A}_k T_s \tag{3.32}$$

This leads to the discrete-time model:

$$\mathbf{x}_{k+1} = \mathbf{A}_{d,k}\mathbf{x}_k + \mathbf{B}_{d,k}\mathbf{u}_k + \mathbf{f}_{0,k} \tag{3.33}$$

where:

$$\mathbf{A}_{d,k} = \mathbf{I} + T_s\mathbf{A}_k \tag{3.34}$$

$$\mathbf{B}_{d,k} = T_s\mathbf{B}_k \tag{3.35}$$

$$\mathbf{f}_{0,k} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k^{(0)}) - \mathbf{A}_k\mathbf{x}_k - \mathbf{B}_k\mathbf{u}_k^{(0)} \tag{3.36}$$

The affine term $\mathbf{f}_{0,k}$ ensures that the discrete-time linear model reproduces the nonlinear dynamics in a very precise way at the linearization point, providing consistency between the original nonlinear system and its discrete-time approximation.

For applications requiring higher accuracy, higher-order integration methods can be employed. The Runge-Kutta family of methods represents a systematic approach to achieving improved accuracy through multiple function evaluations per time step.

For this work, the fourth-order Runge-Kutta (RK4) method is suggested as it is considered to have a good balance between accurate system representation and computational speed. It is able to evaluate the system dynamics at multiple points within each time interval to achieve fourth-order accuracy 3.6.

For the nonlinear system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, the RK4 method computes:

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{3.37}$$

$$\mathbf{k}_2 = \mathbf{f}(\mathbf{x}_k + \frac{T_s}{2}\mathbf{k}_1, \mathbf{u}_k) \tag{3.38}$$

$$\mathbf{k}_3 = \mathbf{f}(\mathbf{x}_k + \frac{T_s}{2}\mathbf{k}_2, \mathbf{u}_k) \tag{3.39}$$

$$\mathbf{k}_4 = \mathbf{f}(\mathbf{x}_k + T_s\mathbf{k}_3, \mathbf{u}_k) \tag{3.40}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{T_s}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{3.41}$$

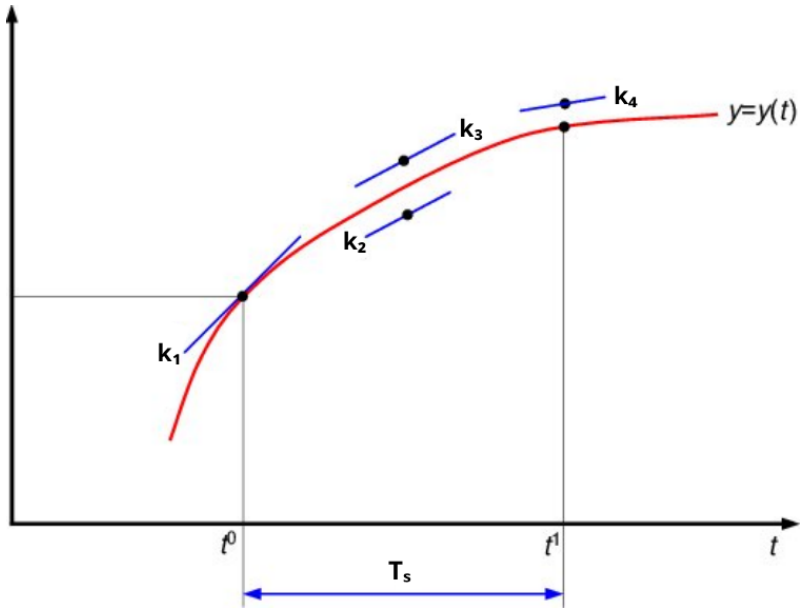


Figure 3.6: This figure shows the 4 slopes $k1$, $k2$, $k3$ and $k4$ used to calculate the final solution. This figure is an adaption of one provided by Flores and Hamid in [15]

This improved accuracy comes at the cost of increased computational complexity, requiring four function evaluations per time step compared to one for the Euler method, for example. However, due to the requirements of marine vehicles, which require high precision and fast dynamics, RK4 method is still justifiable.

Optimization Problem Formulation

The heart of Model Predictive Control lies in the systematic formulation of an optimization problem that encodes control objectives, system constraints, and performance criteria into a mathematical framework suitable for numerical solution. The challenge in MPC design is to construct this optimization problem such that its solution provides control actions that achieve desired system behavior while respecting physical limitations and safety requirements.

The optimization problem formulation process involves :

- Designing the cost function structure, which encodes the relative importance of different control objectives
- Formulating the system constraints, which ensure that physical limitations and safety requirements are respected
- Defining prediction and control horizons, which determine the temporal scope of the optimization

Therefore, the structure of the optimization problem begins with the selection of decision variables. These variables represent the control inputs over the prediction horizon, providing the optimizer with the degrees of freedom needed to achieve desired system behavior.

For efficiency, its made a clear distinction between the prediction horizon N and the control horizon N_c. While the system behavior is predicted over N time steps to ensure adequate foresight, optimization variables are only defined over the shorter control horizon N_c.

Equation 3.42 represents a general nonlinear OCP formulation where a better representation of N and Nc distinction can be observed. This approach recognizes that control predictions become increasingly uncertain over longer time horizons, but still provide important information to incorporate on the problem.

$$\begin{aligned} \min_{U} \quad & J(U, x_0, x_d) = \sum_{k=0}^{N} \left[c^T z_k + z_k^T Q z_k + \phi(z_k) \right] \\ \text{subject to:} \quad & x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots, N-1 \\ & x_0 = x_{\text{measured}} \\ & u_{\min} \leq u_k \leq u_{\max}, \quad k = 0, 1, \ldots, N_c - 1 \\ & g(x_k, u_k) \leq 0, \quad k = 0, 1, \ldots, N-1 \\ & h(x_N) \leq 0 \end{aligned} \tag{3.42}$$

The decision variable vector is structured as:

$$\mathbf{U} = [\mathbf{u}_0^T, \mathbf{u}_1^T, \ldots, \mathbf{u}_{N_c-1}^T]^T \in \mathbb{R}^{3N_c} \tag{3.43}$$

Beyond the control horizon, the control input is held constant at the final optimized value:

$$\mathbf{u}_k = \mathbf{u}_{N_c-1}, \quad \mathbf{k} = \mathbf{N_c}, \ldots, N-1 \tag{3.44}$$

This control horizon implementation provides an important advantage as it reduces the number of optimization variables, typically providing a three-to-four-fold reduction in problem size improving computational efficiency.

In our problem optimization problem is parameterized by current system information and reference trajectories:

$$\mathbf{p} = [\mathbf{x}_0^T, \mathbf{x}_d^T, \omega^T]^T \tag{3.45}$$

where $\mathbf{x}_0$ represents the current measured state, $\mathbf{x}_d$ contains reference state information provided by the guidance system, and ω contains adaptive weight parameters that allow the optimization objective to be adjusted based on operating conditions.

3.4 Global Planning

3.4.1 Map construction

Map extraction

To have a realist map of the real-world scenario, in this case, Leça da Palmeira harbor, within our simulation framework, we begin by generating an occupancy map that captures both static infrastructure (piers, breakwaters) and navigable water.

For this, first, it was acquired a satellite representation of the harbor[1] which was then processed. To estimate real-world distances from the image, the meters-per-pixel resolution at the docking location was calculated using the Web Mercator projection formula:

$$\text{Resolution} = \frac{156543.03 \cdot \cos(\phi)}{2^z} \tag{3.46}$$

Where ϕ is the latitude in radians and z is the zoom level. For the selected location ($\phi = 41.2°$) and a zoom level of 18, the resulting resolution is approximately:

$$\text{Resolution} \approx \frac{156543.03 \cdot \cos(41.2°)}{2^{18}} \approx 0.4497 \text{ meters per pixel} \tag{3.47}$$

This resolution was used to convert pixel measurements on the map to real-world metric units that reflect the true dimensions of the static infrastructure for accurate trajectory planning, obstacle sizing, and control validation.

Then by taking advantage of OpenCV's [2]color space transformations, we isolate the characteristic blue-green hues of open water via adaptive thresholding in HSV space, yielding a binary mask where free (0) and occupied (1) cells are clearly delineated.

Subsequently, the cleaned binary mask undergoes contour extraction; each contiguous obstacle region is converted into a closed polyline, which we then approximate with vertex reduction using the Douglas–Peucker algorithm 1. This simplifies distance checks and visualization.

[1] This Map was obtained using OpenStreetMap, a collaborative project that creates and distributes free and editable geographic data from the whole world.

[2] OpenCV (Open Source Computer Vision Library) is a cross-platform, programming library designed for real-time computer vision and image processing. It provides optimized implementations of common algorithms such as filtering, contour detection, and geometric transformations.

Algorithm 1 Douglas–Peucker Line Simplification

Require: Sequence $P = \{p_1, \ldots, p_n\}$, tol. ε
Ensure: Simplified sequence P'
if $|P| \leq 2$ **then**
  return P
end if
$d_{\max} \leftarrow 0$, $i_{\max} \leftarrow 0$
for $i = 2$ **to** $|P| - 1$ **do**
  $d \leftarrow \text{PerpDist}(p_i, p_1 p_n)$
  if $d > d_{\max}$ **then**
    $d_{\max} \leftarrow d$, $i_{\max} \leftarrow i$
  end if
end for
if $d_{\max} > \varepsilon$ **then**
  split P at $i_{\max}$, recurse on both halves
  return concatenate results
else
  return $\{p_1, p_n\}$
end if

The complete pipeline for the map construction ends with the dilation of the real-sized binary map. This step is very common in this type of works since generally in environments where dimensions are in the 10^2 order, a dilation of 1 meter, for example, does not represent much in terms of extra size, but it has a significant impact when vehicles are navigating near obstacles and extra safety is needed.

Figure 3.7 shows the complete pipeline for map construction. Blue lines shows a zoom vision of a specific zone and how it changes through map alterations and and in the obstacle dilation image, thin red line shows the initial map boundaries before dilation.

3.4.2 Path Planning

To ensure a smooth and predictable behavior from the ASV, it was necessary to provide the ASV with, not only initial and final position and heading, but also with a target state, calculated in the guidance module. Feeding the controller with only the initial and final state the controller is prone to make the optimized trajectory fall in local minima, even with accurate repulsion fields.

To obtain the target state from the guidance module, this must be provided with a trajectory that connects initial and final positions. That trajectory is defined in the grid search module, which, given an initial and final position, executes an algorithm to find an optimized trajectory (a sequence of nodes that have no obstacle between them and that are connected to the initial position in one side and to the final position in the other).

Grid Search

For a discrete map and graph search approach several questions might arise:

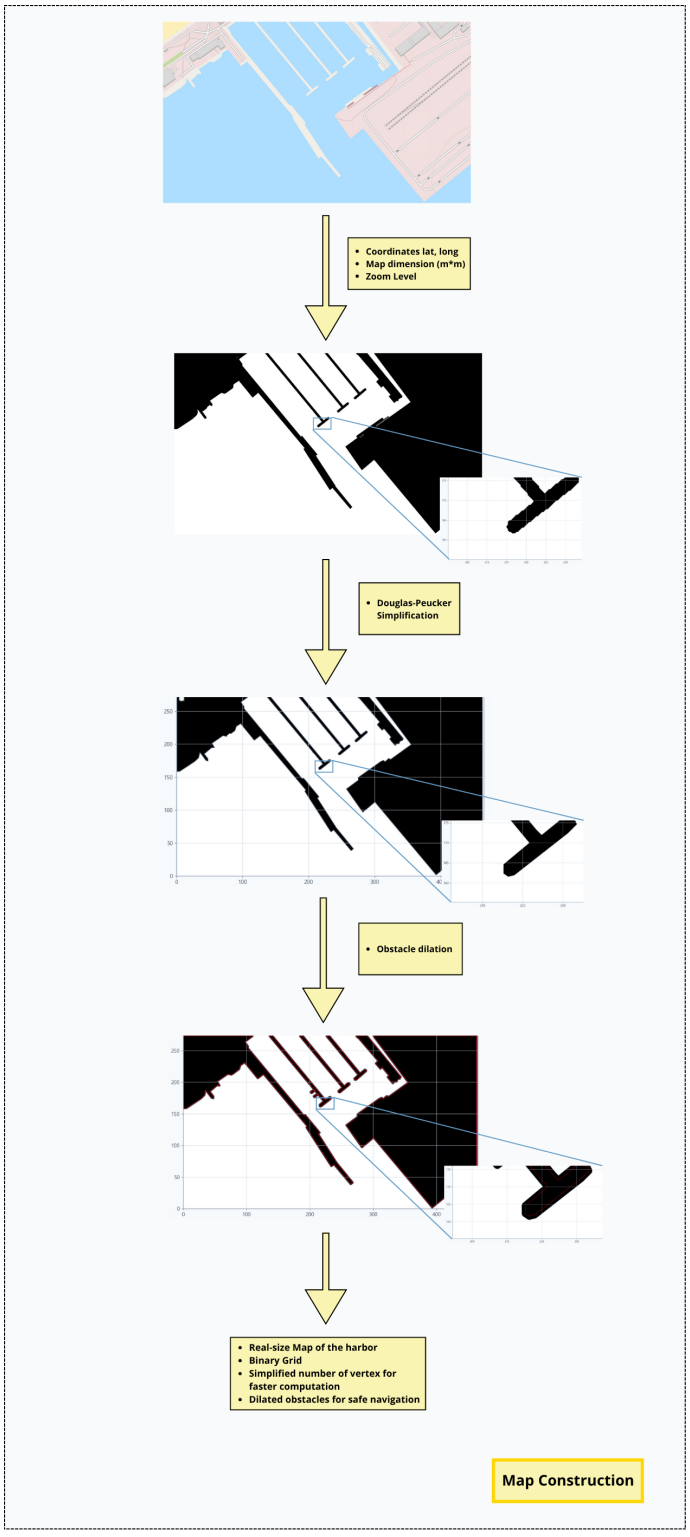


Figure 3.7: Map construction complete pipeline

- What if the environment representation isn't discrete?
- Path consists of straight lines, do we really need the cells in between?

- With large environment size or high dimensions, do graph search algorithms remain feasible, do we need map decomposition?

This work suggests an approach where decomposition may not be needed, specially in high dimension maps, which the case for most of the harbors. So initially an approach using RRT* was tested but it was taking around 30 sedonds to find a path so, a bidirectional extension of RRT 2 was used for faster convergence in large or cluttered harbors, since starting point which can possible be an open water space might haver higher steps but in the docking position side smaller steps are need to "get out" of the docking zone.

In this adaptation, two trees grow simultaneously from the start x_{init} and goal x_{goal}:

$$T_s \leftarrow \{x_{\text{init}}\}, \quad T_g \leftarrow \{x_{\text{goal}}\} \tag{3.48}$$

At each iteration, the algorithm performs the following steps:

1. Sample a random configuration x_{rand}
2. Attempt to connect x_{new} to the nearest node in T_g
3. If collision-free, merge trees
4. If not, swap roles of T_s and T_g and repeat

Algorithm 2 Asymmetric Bidirectional RRT Algorithm

Require: x_{start}, x_{goal}, obstacles, η, N
Ensure: Path or FAIL
$T_s \leftarrow \{x_{start}\}$, $T_g \leftarrow \{x_{goal}\}$
for $i = 1, 2, \ldots, N$ **do**
  $x_{rand} \leftarrow$ Sample()
  $x_{near} \leftarrow$ Nearest(T_s, x_{rand})
  $x_{new} \leftarrow$ Steer$(x_{near}, x_{rand}, \eta)$
  if CollisionFree(x_{near}, x_{new}) **then**
    $T_s \leftarrow T_s \cup \{x_{new}\}$
  end if
  $x_{near} \leftarrow$ Nearest(T_g, x_{rand})
  $x_{new} \leftarrow$ Steer$(x_{near}, x_{rand}, \eta)$
  if CollisionFree(x_{near}, x_{new}) **then**
    $T_g \leftarrow T_g \cup \{x_{new}\}$
  end if
  if trees can connect **then**
    return PATH
  end if
end for
return FAIL

The bidirectional approach provides several computational advantages:

- **Faster convergence**: Two trees explore the space simultaneously, reducing the expected time to find a connection.
- **Balanced exploration**: Each tree focuses on its local region, avoiding unnecessary exploration of distant areas
- **Robustness in cluttered environments**: Particularly effective due to the fact that goal is normally in a confined space requiring, benefiting from small steps to converge faster for a zone out of the confined zone.

The connection criterion between trees uses a distance threshold δ_{connect}:

$$\text{Connectable}(x_{\text{new}}, T_g) = \exists x_g \in T_g : \|x_{\text{new}} - x_g\| \leq \delta_{\text{connect}} \text{ and } \text{CollisionFree}(x_{\text{new}}, x_g) \tag{3.49}$$

The final path is constructed by merging the optimal paths from both trees:

$$\pi^* = \text{Path}(x_{\text{init}} \to x_{\text{connect}}) \oplus \text{Path}(x_{\text{connect}} \to x_{\text{goal}}) \tag{3.50}$$

where x_{connect} is the connection point between the two trees, and $\oplus$ denotes path concatenation.

Path Smoothing

Given a discrete sequence of waypoints $\{P_0, P_1, \ldots, P_M\}$ from the BI-RRT* grid search algorithm, we fit a cubic B-spline curve $s(\tau) \in \mathbb{R}^2$, $\tau \in [0,1]$, defined by:

$$s(\tau) = \sum_{i=0}^{M+2} N_{i,3}(\tau) P_{i-1} \tag{3.51}$$

where $N_{i,3}(\tau)$ are the degree-3 B-spline basis functions on a uniformly spaced knot vector. The basis functions are defined recursively as:

$$N_{i,0}(\tau) = \begin{cases} 1, & \tau_i \leq \tau < \tau_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{3.52}$$

$$N_{i,k}(\tau) = \frac{\tau - \tau_i}{\tau_{i+k} - \tau_i} N_{i,k-1}(\tau) + \frac{\tau_{i+k+1} - \tau}{\tau_{i+k+1} - \tau_{i+1}} N_{i+1,k-1}(\tau) \tag{3.53}$$

To enforce a path without collision, the WP list near obstacles is densified in order to guarantee that the interpolation curve fits the desired direction.

The spline parameterization provides several advantages for ASV trajectory following:

- **Smoothness**: The cubic B-spline ensures C^2 continuity, providing smooth velocity and acceleration profiles
- **Local control**: Modifications to individual WP only affect a local region of the curve

- **Computational efficiency**: Spline evaluation requires only local basis function computation

The resulting spline is smooth, stays within the convex hull of the WPs, and can be evaluated efficiently for guidance. Figure 3.8 shows an example of the spline path plotted in the simulator, clearly showing that it reaches all reference WPs, still keeping its continuity and smoothness.

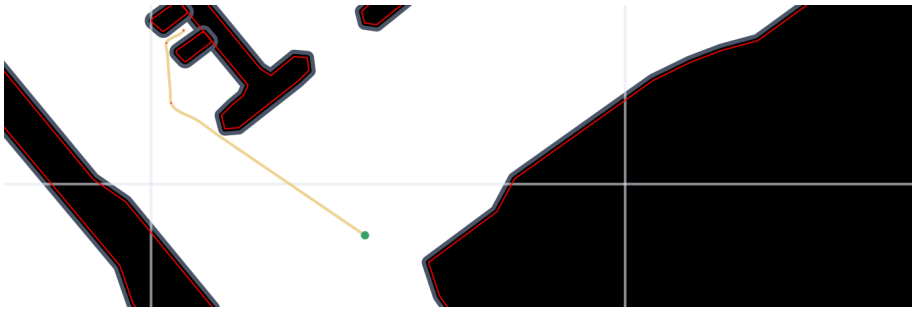


Figure 3.8: Spline path is shown in light orange. Small red dots identify the WPs found by the grid search algorithm.

3.5 Guidance and Control Framework

3.5.1 Guidance System Structure

The main role of the Guidance module is to provide the MPC with the reference state for which the MPC optimization will be aimed to.

The system flow shown in 3.9 is followed :

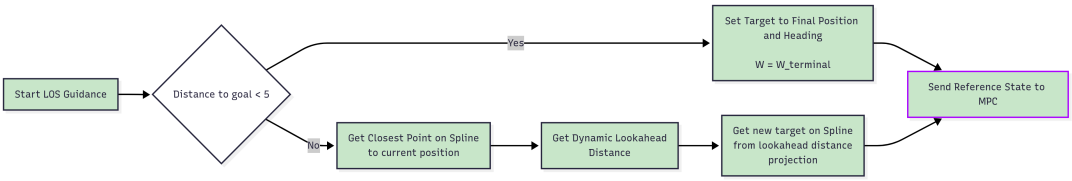


Figure 3.9: Guidance system diagram

Spline-Based Path Representation

The foundation of our guidance system relies on a smooth path representation using cubic splines. Rather than following discrete WPs, the desired trajectory is parameterized as a continuous curve using arc-length parameterization:

$$\mathbf{p}(s) = [x(s), y(s)]^T \tag{3.54}$$

where $s \in [0, s_{total}]$ represents the arc-length parameter along the spline path. This parameterization offers several advantages over discrete waypoint following:

1. **Smoothness**: Eliminates discontinuities in curvature that can cause control oscillations
2. **Precision**: Enables exact calculation of path properties such as curvature and tangent direction
3. **Flexibility**: Allows for smooth path modifications and re-planning
4. **Mathematical convenience**: Facilitates precise lookahead distance calculations

After getting the computed spline from the global planning module, the closest point from the current position to the spline is calculated, and from that point, the dynamic LOS algorithm determines the "lookahead" point in arc-length distance s. The dynamic "lookahead" calculation follows the following equation 3.59, creating a LOS guidance system.

Dynamic LOS Strategy

LOS guidance, also known as Pure Pursuit or 'carrot chasing', represents a fundamental approach to path-following control for autonomous vehicles. The core principle involves continuously guiding the vehicle toward a target point positioned at a fixed lookahead distance along the desired path. It begins with determining the vessel's current position relative to the desired path. The current path parameter is found by solving the closest point projection:

$$s_{current} = \arg\min_{s} \|\mathbf{p}(s) - \mathbf{p}_{current}\| \tag{3.55}$$

The LOS target point is then computed by advancing a fixed lookahead distance L_0 along the path:

$$\dot{s} = s_{current} + L_0 \tag{3.56}$$

$$\mathbf{p}_{LOS} = \mathbf{p}(s_{target}) = [x_{LOS}, y_{LOS}]^T \tag{3.57}$$

The defined heading angle steers the vessel directly toward this target point:

$$\mathbf{X}_{ref} = \begin{bmatrix} x_{ref} \\ y_{ref} \\ \psi_{ref} \end{bmatrix} = \begin{bmatrix} x_{LOS} \\ y_{LOS} \\ \arctan 2(y_{LOS} - y_{current}, x_{LOS} - x_{current}) \end{bmatrix} \tag{3.58}$$

While this classical approach provides stable path-following behavior, the fixed lookahead distance L_0 may lead to stagnation or hard-to-tune speed vs position control.

Our guidance system is an adaptation to optimize the classic lookahead strategy :

We implement a dynamic feedback mechanism that responds to the vessel's tracking performance. The system monitors the distance to the previous LOS target and adjusts the lookahead distance accordingly:

$$L_0 = \begin{cases} (v_{ref} - \alpha \cdot d_{target}) \cdot Ts & \text{if } d_{target} < 5 \text{ m} \\ 2 & \text{otherwise} \end{cases} \tag{3.59}$$

where T_s is the sampling time and $d_{target} = \|\mathbf{p}_{vessel} - \mathbf{p}_{LOS,prev}\|$ represents the distance to the previous LOS target, $\alpha = 0.08$ is the distance feedback gain, and $v_{ref} = 0.3$ m/s provides a velocity-dependent component.

This dynamic adjustment implements a simple yet effective control law: when the vessel approaches the target point closely ($d_{target} < 5$ m), the lookahead distance increases to maintain a smooth trajectory generation. To ensure a non-stagnated lookahead, an extra safety non-decreasing constraint was added.

$$L_0^{k+1} = \max(L_0^{k+1}, L_0^k) \tag{3.60}$$

Context-Based Adaptive Weight Selection

The effectiveness of multi-objective cost functions depends critically on appropriate weight selection that reflects the relative importance of different objectives. Static weight selection often performs poorly across diverse operating conditions, as the appropriate balance between objectives varies significantly with factors such as proximity to obstacles, distance from target waypoints, and mission phase.

The adaptive weight selection strategy addresses this challenge by automatically adjusting cost function weights based on current operating conditions:

$$\omega_k = \begin{cases} \omega_{tracking} & \text{if } \|\mathbf{e}_{pos,k}\| > d_{threshold} \\ \omega_{terminal} & \text{if } \|\mathbf{e}_{pos,k}\| \leq d_{threshold} \end{cases} \tag{3.61}$$

where $\mathbf{e}_{pos,k} = [\mathbf{x}_k]_{xy} - [\mathbf{x}_d]_{xy}$ represents the position tracking error.

During approach phases, when the vehicle is far from its target, the weights $\omega_{tracking}$ emphasize trajectory tracking and efficient progress toward the goal. These weights typically assign high importance to position tracking while allowing moderate control effort to achieve rapid convergence.

As the vehicle approaches the target position (within distance $d_{threshold}$), the weights shift to $\omega_{terminal}$, which emphasize precision and stability. These terminal weights typically reduce the relative importance of rapid tracking in favor of smooth control actions and precise final positioning.

For the purpose of this work, the weight values are defined as follows :

$$\omega_{\text{tracking}} = [10,\ 0.5,\ 0,\ 0,\ 100], \quad \omega_{\text{terminal}} = [5,\ 30,\ 1.5,\ 1,\ 200].$$

where,

$$\omega = [\omega_{pos}, \omega_{\psi}, \omega_{v_x}, \omega_{v_y}, \omega_r]$$

There are also terms for penalizing thrusters directly. The most crucial is w_{thr} which is used to penalize the difference between these values (in N) to zero. It takes the following form :

$$\omega_{thr} = [\omega_{th1}, \omega_{th2}, \omega_{th3}]$$

During movement, w_{thr} values also change, similarly to w_k but for different functionalities.

$$\omega_{thr} = \begin{cases} [0,0,1000] & \text{if } \|\mathbf{e}_{pos,k}\| > d_{threshold} \\ [0,0,0] & \text{if } \|\mathbf{e}_{pos,k}\| \leq d_{threshold} \end{cases} \tag{3.62}$$

Penalizing tunnel thruster input (lateral movement) during homing and stopping penalization while in docking motion since lateral motion is crucial for it.

$w_{smooth} = 0.1$ is also used to penalize the difference between two consecutive thruster inputs, to promote a smooth and stable movement.

This adaptive strategy addresses a fundamental challenge in marine vehicle control: the same controller must perform effectively across dramatically different operating regimes, from high-speed transit in open water to precision maneuvering in confined spaces near docks. The weight adaptation provides a systematic mechanism for achieving appropriate behavior across these diverse conditions without requiring manual parameter adjustment or mode switching.

The implementation of adaptive weight selection requires careful consideration of the transition between weight sets. Abrupt changes in weights can cause discontinuities in the optimization problem, potentially leading to poor performance or convergence difficulties. Smooth transition functions, such as sigmoid or linear interpolation over a small region around the threshold, can help maintain optimization stability while providing the desired adaptive behavior.

3.5.2 MPC

3.5.2.1 Obstacle Avoidance Formulation

Map Interaction

In order to interact with the given map, to obtain the distances between the ASV and the surrounding obstacles, three hypotheses come up.

- Represent the vehicle as a particle and dilate the obstacle to at least the longer size of the vehicle for safety
- Represent the vehicle as a circle with a diameter of at least the longer side of the vehicle, usually more for safety
- Represent the vehicle as an ellipse or superellipse (also known as Laimé curves).

For this work it was chosen to combine first and second methods, for fast computation and extra safety, although it recognizes that last method would be the more realistic.

Obstacles Representation

Initial Approach : Obstacles as superellipses

Initially it was tested in simulation a version for the dock where obstacles where superellipses. These were chosen due to being differentiable, easy to test and most of all,

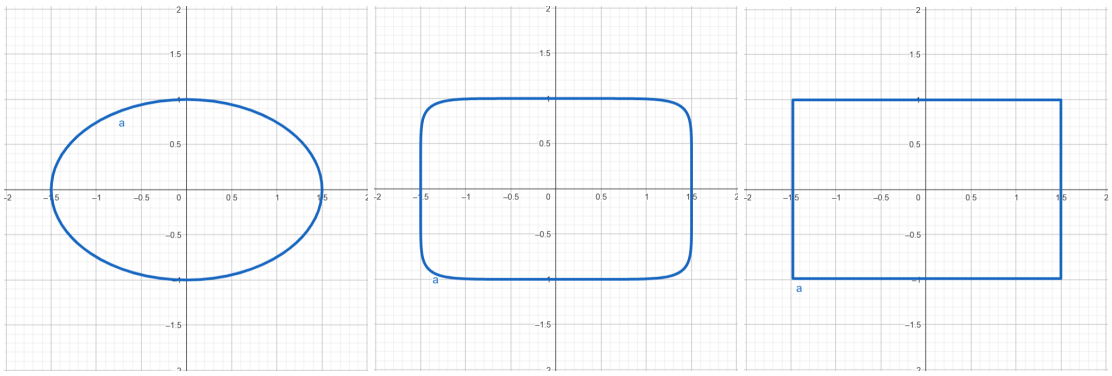
The function of an ellipse is :

$$\left|\frac{x}{a}\right|^2 + \left|\frac{y}{b}\right|^2 - 1 = 0, \tag{3.63}$$

A superellipse comes from the classic ellipse function but with a tunable power term, instead of fixed 2. 3.64 presents the general superellipse equation :

$$\left|\frac{x}{a}\right|^{\mathbf{n}} + \left|\frac{y}{b}\right|^{\mathbf{n}} - 1 = 0, \tag{3.64}$$

The bigger the power term, the sharper the edges will be, and the more like a rectangle it will look like. Figure below shows a comparison between the power terms, with the realistic version of the ASV length and width (1.5m and 1m respectively). Subfigure 3.10a shows a classic ellipse, where n = 2. Subfigure 3.10b shows a superellipse where n = 8 and subfigure 3.10c shows a version where n = 1000.



(a) $n = 2$ (b) $n = 8$ (c) $n = 1000$

Figure 3.10: Visualization of power term variation in Superellipses

For the purpose of this work, it was used the version where n=8 as it was a good trade-off between computational complexity and realistic simulation.

After this, it was possible to define the distance to the obstacle as the distance from the vehicle's center of mass to the border of the superellipse.

$$\left|\frac{x}{a}\right|^n + \left|\frac{y}{b}\right|^n - 1 \leq 0, \tag{3.65}$$

where :

$$a = \frac{L}{2}, \quad b = \frac{W}{2}, \quad n > 2, \quad (x,y) \in \mathbb{R}^2.$$

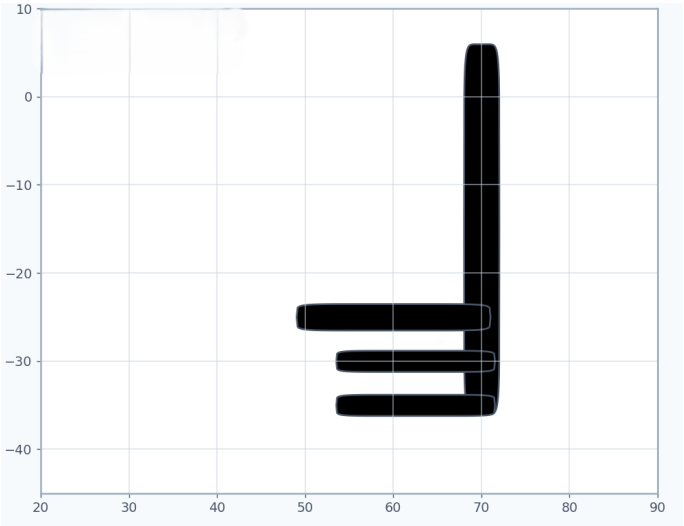
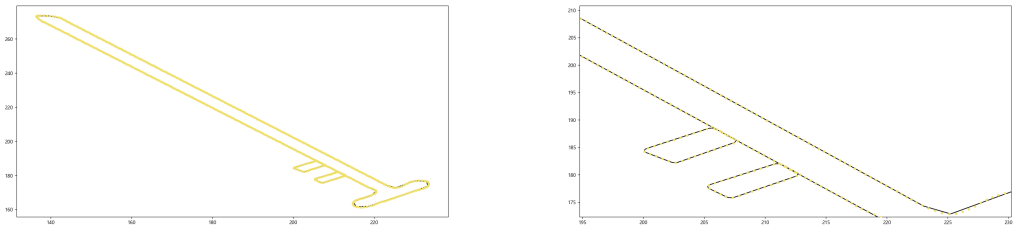


Figure 3.11: Initial Map using obstacles modulated as superellipses

Final Approach : Point-Based Obstacle Avoidance with Hard Constraints

For the final version, it was the extracted map from the image of the harbor which was then processed and turned into a binary map where docks are polygons, which were then simplified using the Douglas-Peucker algorithm and then dilated for safety.

For obstacle avoidance in the MPC formulation, we employ a point-based discretization approach that converts the simplified polygon obstacles into densely distributed boundary points, as seen in figure 3.12.



(a) Dock visualization (b) Zoomed dock visualization

Figure 3.12: Densified polygon line segments for obstacle avoidance

Polygon Discretization

Each polygon obstacle $\mathbf{O}_i$ with vertices $\{v_{i,1}, v_{i,2}, \ldots, v_{i,m_i}\}$ is discretized by densifying each edge segment. For edge $e_{i,j}$ connecting vertices $v_{i,j}$ and $v_{i,j+1}$, we generate N_{dense} uniformly distributed points:

$$\mathbf{p}_{i,j,k} = v_{i,j} + \frac{k}{N_{\text{dense}}}(v_{i,j+1} - v_{i,j}), \quad k = 0, 1, \ldots, N_{\text{dense}} \tag{3.66}$$

This creates a dense point set $P_i = \{\mathbf{p}_{i,j,k}\}$ representing the boundary of the obstacle $\mathbf{O}_i$.

Distance Calculation

The minimum distance from the ASV position $\mathbf{p}_{\text{ASV}} = [x, y]^T$ to obstacle $\mathscr{O}_i$ is computed as:

$$d(\mathbf{p}_{\text{ASV}}, \mathbf{O}_i) = \min_{\mathbf{p} \in \mathbf{P}_i} \|\mathbf{p}_{\text{ASV}} - \mathbf{p}\|_2 - \frac{L_{\text{ASV}}}{2} - d_{\text{safety}} \tag{3.67}$$

where:

- L_{ASV} is the characteristic length of the ASV
- d_{safety} is an additional safety margin

Hard Constraint Formulation

The collision avoidance requirements are enforced through hard constraints in the MPC optimization problem:

$$d(\mathbf{x}^{xy}_{k+i|k}, \mathbf{O}_j) \geq d_{\min}, \quad \forall i \in \{0, 1, \ldots, N-1\}, \quad \forall j \in \{1, \ldots, N_{\text{obs}}\} \tag{3.68}$$

where:

- $\mathbf{x}^{xy}_{k+i|k} = [x_{k+i|k}, y_{k+i|k}]^T$ represents the predicted position at time step $k+i$
- $d_{\min} > 0$ is the minimum allowable distance to obstacles
- N_{obs} is the total number of obstacles

Computational Considerations

The point densification parameter N_{dense} presents a trade-off between:

- **Accuracy**: Higher N_{dense} provides more precise boundary representation
- **Computational efficiency**: Lower N_{dense} reduces the number of distance calculations per optimization iteration

For real-time applications, we typically choose $N_{\text{dense}} = 10$ points per meter of edge length, providing sufficient resolution while maintaining computational tractability.

The discretization error is bounded by $\varepsilon_{\text{disc}} \leq \frac{L_{\text{edge}}}{2N_{\text{dense}}}$, where L_{edge} is the maximum edge length, ensuring that the approximation quality can be controlled through an appropriate choice of N_{dense}.

3.5.3 Cost Function Design

Multi-Objective Cost Function Design

The cost function represents the mathematical encoding of control objectives and performance criteria. In real-world applications, multiple competing objectives must be balanced, requiring careful design of the cost function structure to achieve appropriate trade-offs between different performance goals.

The temporal structure of MPC naturally leads to a cost function that evaluates performance over the entire prediction horizon:

$$J(\mathbf{x}_k, \mathbf{U}_k) = \sum_{k=0}^{N-1} L(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_d, \omega) + V_f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_d, \omega) \tag{3.69}$$

The stage cost function $L(\cdot)$ evaluates performance at each prediction step, while the terminal cost function $V_f(\cdot)$ provides additional emphasis on final state accuracy.

$$V_f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_d, \omega) = \sum_{k=0}^{N-1} \omega_v \cdot L(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_d, \omega) \tag{3.70}$$

Where ω_v represents the final weight emphasis variable.

This structure allows the optimization to consider both transient behavior and steady-state performance.

For autonomous surface vehicle applications, the stage cost function must balance several competing objectives:

$$\begin{aligned} L(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_d, \omega) = {} & \omega_{\text{pos}} \left\| [x_k, y_k]^\top - [x_d, y_d]^\top \right\|^2 \\ & + \omega_{v_x} (v_{x,k} - v_{x,d})^2 + \omega_{v_y} (v_{y,k} - v_{y,d})^2 \\ & + \omega_\psi (\psi_k - \psi_d)^2 + \omega_r (r_k - r_d)^2 \\ & + \omega_{\text{smooth}} \left\| \mathbf{u}_k - \mathbf{u}_{k-1} \right\|^2 + \sum_{i=1}^{3} \omega_{thr_i} u_{i,k}^2 \end{aligned} \tag{3.71}$$

with $[v_x, v_y, r] = [u, v, r]$, body frame fixed, velocities.

The meaning of each penalization term is as described below :

- The position tracking term $\omega_{pos} \| [\mathbf{x}_k]_{xy} - [\mathbf{x}_d]_{xy} \|^2$ penalizes deviations from the desired target. The notation $[\mathbf{x}_k]_{xy}$ indicates the position components of the state vector, specifically the x and y coordinates. The weight ω_{pos} determines the relative importance of tracking the desired target.

- The heading tracking term $\omega_{\psi}\,(\psi_k - \psi_d)^2$ serves the same purpose as position penalization, but for angle. This allows to align the vehicle with the target along the homing motion and most importantly, allows set a desired final angle, crucial for docking.
- The velocity tracking terms $\omega_{v_x}\,(v_{x,k} - v_{x,d})^2$ and $\omega_{v_y}\,(v_{y,k} - v_{y,d})^2$ serve multiple purposes the control system. When reference velocities are provided by the guidance system, this term enforces desired speed profiles. When reference velocities are set to zero, this term provides damping to reduce oscillatory behavior and improve stability. These appear separated by axis due to being more versatile if condition-specific tuning is needed.
- The heading rate tracking term $\omega_{r}\,(r_k - r_d)^2$ allows to reduced heading and consequently movement oscillation while the vehicle is moving. For this term, the r_d is always set to 0, since the aim is to reduce unnecessary oscillations.
- $\omega_{smooth}\|\mathbf{u}_k - \mathbf{u}_{k-1}\|^2$ serves as a smoothing term, to control the thruster's force variation. This is used in order to prevent abrupt changes that could cause unrealistic real-life movements or even cause the the controller to lose stability.
- $\omega_{thr_i}\,u_{i,k}^2$ also penalizes thrusters directly. In this work, funnel thruster has a higher during homing since it's not really needed.

Complete MPC Formulation with Obstacle Constraints

The MPC optimization problem becomes:

$$
\begin{aligned}
\min_{\mathbf{U}_k} \quad & J_k(\mathbf{U}_k, \mathbf{x}_k) \\
\text{s.t.} \quad & \mathbf{x}_{k+i+1|k} = \mathbf{A}_{d,k}\mathbf{x}_{k+i|k} + \mathbf{B}_{d,k}\mathbf{u}_{k+i|k} + \mathbf{f}_{d,k}^{(0)} \\
& \mathbf{x}_{k|k} = \mathbf{x}_k \\
& \mathbf{u}_{\min} \leq \mathbf{u}_{k+i|k} \leq \mathbf{u}_{\max} \\
& d(\mathbf{x}_{k+i|k}^{xy}, \mathbf{O}_j) \geq d_{\min}, \quad \forall i, j \\
& i = 0, 1, \ldots, N_c - 1
\end{aligned}
\tag{3.72}
$$

where $J_k(\mathbf{U}_k, \mathbf{x}_k)$ is the one mentioned in3.69 and constraints are as defined in 3.68.

OCP Solving

At each sampling instant t_k, the MPC controller solves the following finite-horizon optimal control problem:

$$\min_{\mathbf{U}_k} \quad J_k(\mathbf{U}_k, \mathbf{x}_k) \tag{3.73}$$

$$\text{s.t.} \quad \mathbf{x}_{k+i+1|k} = \mathbf{A}_{d,k}\mathbf{x}_{k+i|k} + \mathbf{B}_{d,k}\mathbf{u}_{k+i|k} + \mathbf{f}_{d,k}^{(0)} \tag{3.74}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_k \tag{3.75}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_{k+i|k} \leq \mathbf{u}_{\max}, \quad i = 0, \ldots, N_c - 1 \tag{3.76}$$

$$\mathbf{u}_{k+i|k} = \mathbf{u}_{k+N_c-1|k}, \quad i = N_c, \ldots, N-1 \tag{3.77}$$

where:

- $\mathbf{U}_k = [\mathbf{u}_{k|k}^T, \mathbf{u}_{k+1|k}^T, \ldots, \mathbf{u}_{k+N_c-1|k}^T]^T \in \mathbb{R}^{3N_c}$ is the control sequence
- N is the prediction horizon
- $N_c \leq N$ is the control horizon
- $\mathbf{x}_{k+i|k}$ denotes the predicted state at time $k+i$ given information at time k

Cost Function Design

The objective function incorporates trajectory tracking performance and control effort minimization:

$$J_k(\mathbf{U}_k, \mathbf{x}_k) = \sum_{i=0}^{N-1} L(\mathbf{x}_{k+i|k}, \mathbf{u}_{k+i|k}, \mathbf{x}_{ref}) + V_f(\mathbf{x}_{k+N|k}, \mathbf{x}_{ref}) \tag{3.78}$$

where the stage cost is defined as:

$$L(\mathbf{x}, \mathbf{u}, \mathbf{x}_{ref}) = \|\mathbf{x}_{pos} - \mathbf{x}_{ref,pos}\|_{\mathbf{Q}}^2 + \|\mathbf{u} - \mathbf{u}_{ref}\|_{\mathbf{R}}^2 \tag{3.79}$$

and the terminal cost is:

$$V_f(\mathbf{x}, \mathbf{x}_{ref}) = q_f \|\mathbf{x}_{pos} - \mathbf{x}_{ref,pos}\|_{\mathbf{Q}_f}^2 \tag{3.80}$$

Here, $\mathbf{x}_{pos} = [x, y]^T$ represents the position components, $\mathbf{Q}, \mathbf{Q}_f \succ 0$ are positive definite weighting matrices for state deviations, $\mathbf{R} \succ 0$ is the control effort weighting matrix, and $q_f > 1$ is the terminal cost amplification factor.

3.5.4 Simulation Control Scheme

For simulation and monitoring the following scheme was used 3.13. The controller switches weights when we is 7m from the final target and then a counter k is used to count the number of times the requirements (position error $[x_k, y_k] - [x_d, y_d]$ and heading error $\psi_k - \psi_d$) where satisfied.

Then if they were satisfied 50 times (5 seconds) it is considered as completed and the simulation finishes.

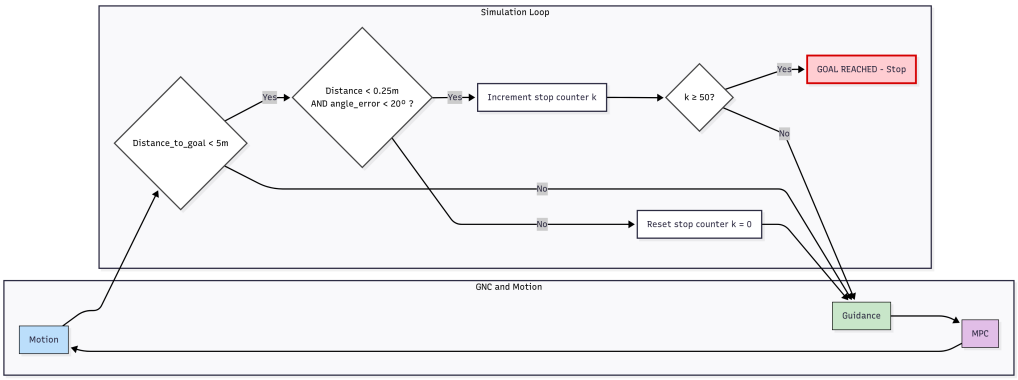


Figure 3.13: Simulation scheme showing the relationship between simulation software and the other systems, defining if simulation continues or not

3.6 Strategies and Computational Tools

Interior Point Optimizer

The optimization problem (3.73)–(3.77) is solved using the Interior Point Optimizer (IPOPT), which implements a primal-dual interior point method. The algorithm converts inequality constraints to equality constraints through the introduction of slack variables and applies a logarithmic barrier approach.

The barrier problem takes the form:

$$\min_{\mathbf{U},\mathbf{s}} \quad J_k(\mathbf{U}) - \mu \sum_j \log(s_j) \tag{3.81}$$

subject to equality constraints, where $\mathbf{s}$ are slack variables and $\mu > 0$ is the barrier parameter.

Automatic Differentiation

Gradient and Hessian computations are performed using CasADi's automatic differentiation capabilities, ensuring exact derivatives without finite difference approximations. This approach significantly enhances convergence rates and numerical stability.

Warm Starting Strategy

To improve computational efficiency, a warm starting technique is employed where the initial guess for the current optimization is constructed by shifting the previous optimal solution:

$$\mathbf{U}_k^{(0)} = [\mathbf{u}_{k|k-1}^{*T}, \mathbf{u}_{k+1|k-1}^{*T}, \ldots, \mathbf{u}_{k+N_c-2|k-1}^{*T}, \mathbf{u}_{k+N_c-1|k-1}^{*T}]^T \tag{3.82}$$

This strategy typically reduces the number of interior point iterations required for convergence, thereby decreasing computational time.

Chapter 4

Experimental Validation and Results

4.1 Introduction

In order to simulate the complete docking process, an user interface (UI) was created using Python, more specifically, *Matplotlib* library, taking advantage of CasADI's seamless integration with Python, creating the possibility to use a less common programming language for control. Once defining (outside of the GUI) the initial position and desired final heading, the GUI shows the final version of the dilated map and allows interactive defining of the docking position with a click on the map. Then it also allows choosing, path following method or direct method (without guidance). Once finished the initial process, the GUI iteratively updates the visualization with the ASV represented by the already mentioned superellipse and its movement through the harbor dilated map.

Figure 4.1 shows the UI used for simulation tests.

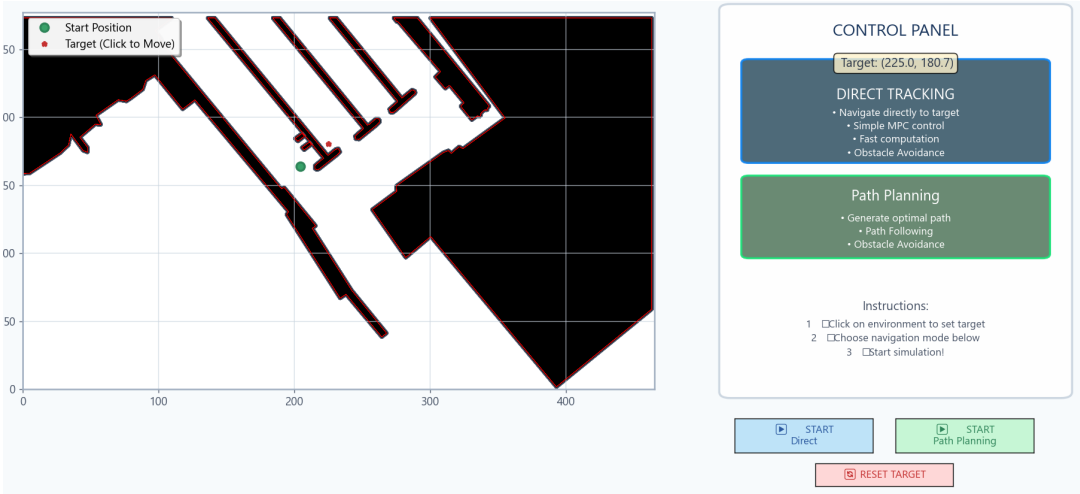


Figure 4.1: UI used for simulation tests

4.2 Preliminary Results

Performance Metrics

Initially, an approach modulating obstacles as superellipses was tested. This approach was a success, not only in standard metrics such as speed, position, heading and stabilization but also computationally, since it proved to be real-time feasible.

One of the first completed docking movements captured in simulation is showed in figure 4.2. The arrows point towards the vehicles direction and these were plotted for every 20 iterations with $T_s = 0.1s$, so 2 every 2 seconds.

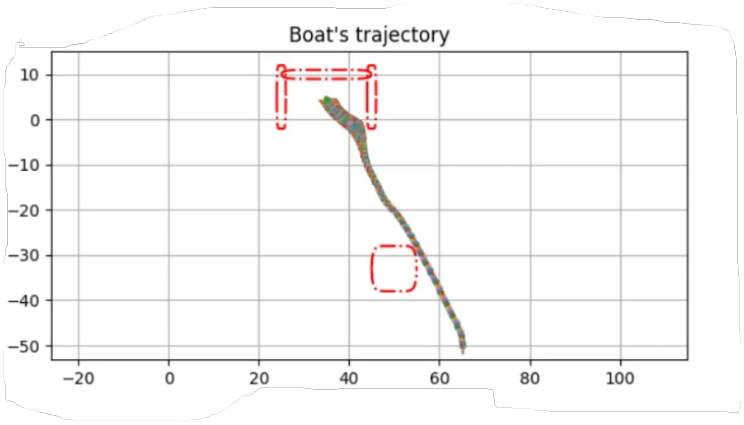
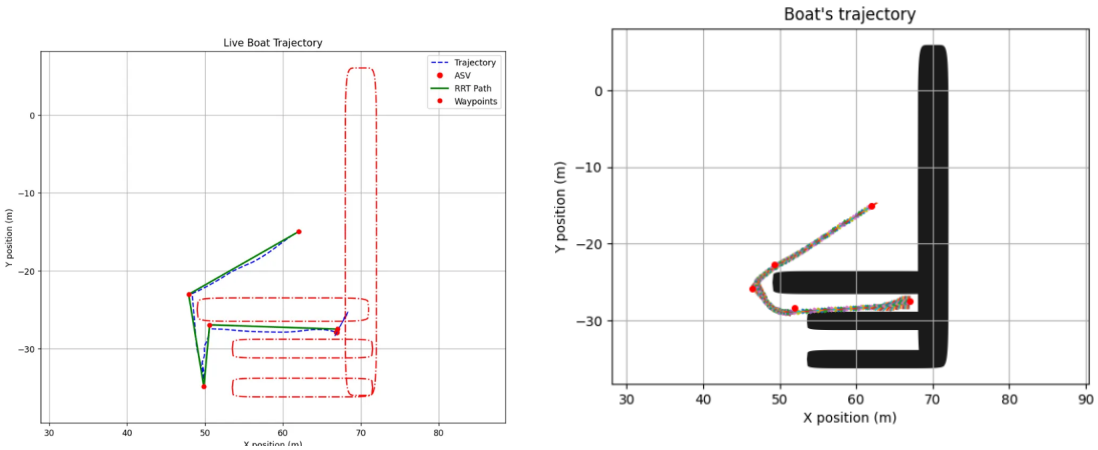


Figure 4.2: In red are the static obstacles. Coloured arrows represent vehicles position and heading in space.

Obstacles are represented through the red dashed line and the vehicle follows an optimal trajectory without hitting any obstacle and being able to change its heading in the end and stabilize in order to stop in the desired final position with the desired heading. In this version, the only reference sent to the MPC was the final point, final heading, and obstacles, no global planning nor guidance was being apply. Obstacle avoidance was modulated through equation mentioned in 3.65 and implemented using hard constraints.

Afterwards, keeping the obstacles as superellipse, a new map was created in order to teste and more complex solutions:



(a) (b)

Figure 4.3: Second version of the map used for simulation using obstacles as superellipes

All simulations presented on this chapter were made using a sampling time $T_s = 0.1s$ a prediction horizon and a $N = 15s$ and a control horizon $5s$. For the weights of the cost function, the weights used were the ones already mentioned in chapter 3

4.3 Final Simulation Environment and Results

The final results were achieved through the realistic harbor scenario and the following section will take advantage of the scenario through the exploitation of the following use cases:

- **Use case PH**: Moving and stabilizing in a target position with a predefined heading
- **Use case PHO**: Moving and stabilizing in a target position with a predefined heading with an obstacle in between
  - Without path planner (only repulsive forces)
  - With path planner and repulsive forces
- **Use case C**: Complete Docking Motion with different obstacles

For each demonstration of the achieved results the **simulation plot** will be followed by:

- Plot with *x*, *y*, and *psi* (heading) overtime, and a dashed line showing the reference value for each.
- Plot with *surge*, *sway*, and *yaw* overtime, with a dashed line showing the reference value for each.
- Plot with thrusters force for each thruster overtime.
- Plot showing CPU solving time for the OCP.

The thruster's and computational time plots won't be mentioned often since, in most cases, there aren't big visual differences between each. The most important thing to notice in the thruster's plot is the capacity of, through the mpc, defining when to use each thruster. In this work, tunnel thruster is barely used until final docking movement as it can be seen in the graphs.

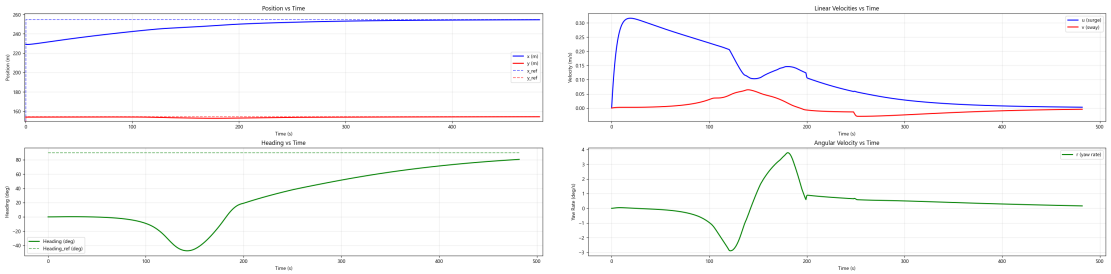
4.3.1 Use case PH

Heading aligned with next position

In image 4.4. it is possible to observe the vehicle (represented by the rounded rectangle) moving towards the target position, rotating and stopping with the desired heading.

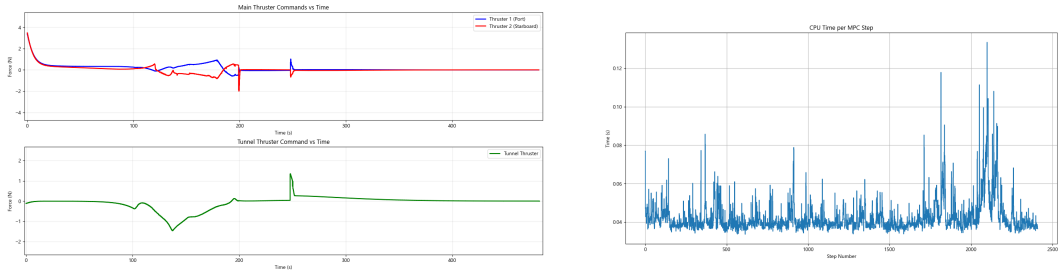


Figure 4.4: Use Case PH (Heading not aligned) : Simulation



(a) (b)

Figure 4.5: Use Case PH (Heading not aligned) : Linear and angular velocities and $[x, y, \psi]$ changing over time.



(a) (b)

Figure 4.6: Use Case PH (Heading not aligned) : Thrusters input variation along time and time to solve the OCP problem

The graphs above 4.5b show how the linear and angular velocities and $[x, y, \psi]$ change over time for this simulation, proving the smooth and stable behaviour of the vehicle

Heading not aligned with final position

To test the feasibility of this use case, the initial angle was defined as an angle not aligned with the final position. Image below **??** show controller is also able to drive the vehicle correctly in this scenarios.

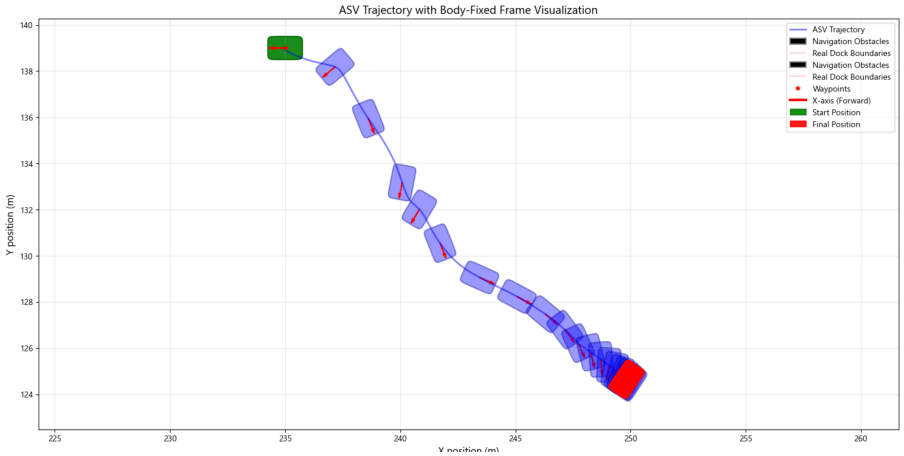
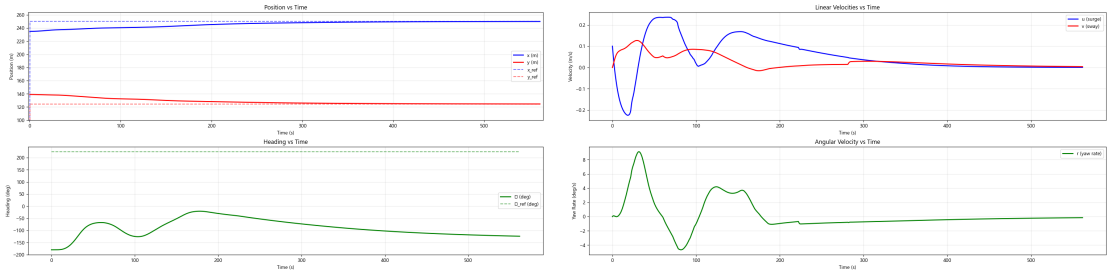
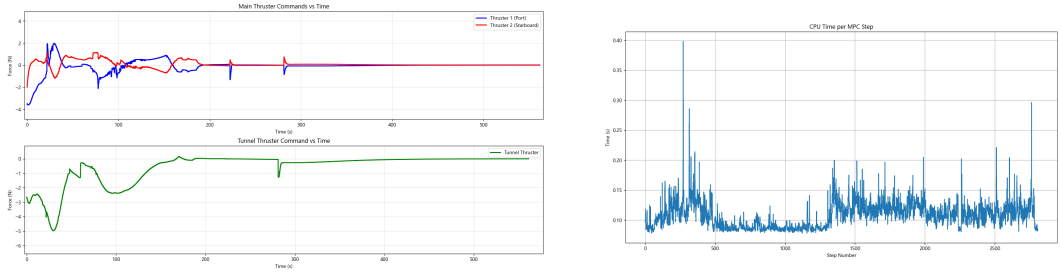


Figure 4.7: Use Case PH (Heading aligned) : Simulation



(a) (b)

Figure 4.8: Use Case PH (Heading aligned) : Linear and angular velocities and $[x, y, \psi]$ changing over time.



(a) (b)

Figure 4.9: Use Case PH (Heading aligned) : Thrusters input variation along time and time to solve the OCP problem.

In the image 4.7 we can visualize a simulation of use case PH but in a case where the initial heading is not aligned with the final position. The controller is capable of making the vehicle rotate, in order to get to the desired heading (aligned with final position) until in is at a distance lower than 5 meters from it, where weights and desired heading change and the vehicle it start rotating and slowing down, in order to achieve the desired final position and heading, in a stable position, without moving.

4.3.2 Use case PHO

This use case differs from the previous one since it already has applies obstacle avoidance but it differs from the last one since there is no guidance. The reference for the MPC is always the desired final position, and only heading changes, since before being close to the final position the target heading is the *arctan*2 between the vehicle's current position and target position and when it is close to the final position the heading change to the one desired for final pose. Weights change when heading changes, as mentioned in 3.61.

It's visible in 4.10 there is no reference waypoint used to generate a path, but obstacle avoidance integration was sufficient to drive the vehicle safely.

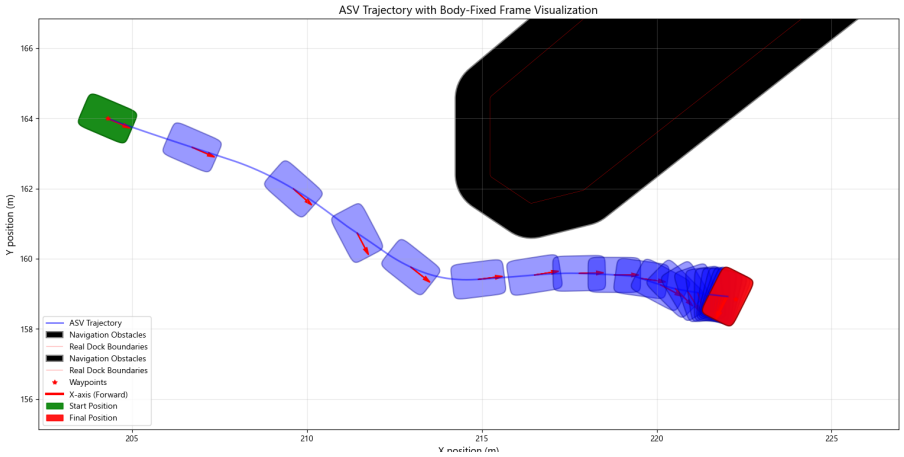
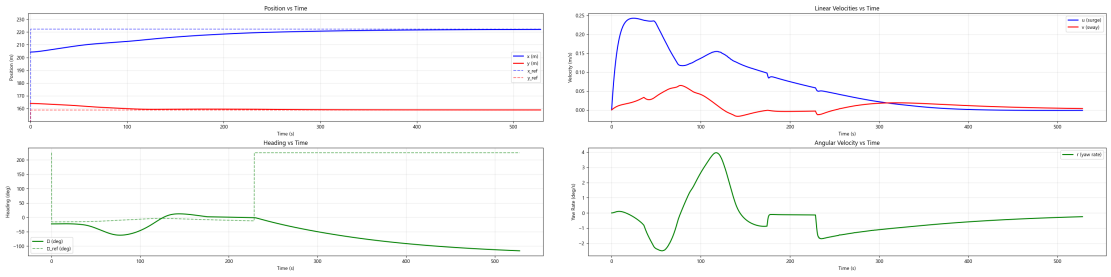
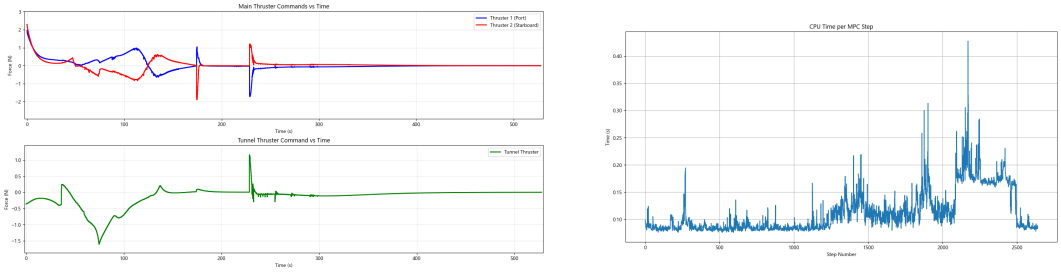


Figure 4.10: Use Case POH : Simulation



(a) (b)

Figure 4.11: Use Case POH : Linear and angular velocities and $[x, y, \psi]$ changing over time.



(a) (b)

Figure 4.12: Use Case POH : Thrusters input variation along time and time to solve the OCP problem.

Graphs above 4.11a-4.11b show, specially in velocities, less smooth variations, with small dumps from time to time, contrarily to graphs from other simulations. This happens due to the obstacle hard constraints inserted in the controller.

Although it worked for this situation, this only works in certain situations where the OCP doesn't fall in local minima or becomes infeasible. It's for that reason that the guidance system

was developed, creating a more controlled scenario. This obstacle avoidance method is computationally expensive is it can be seen in 4.12b, although it can be tuned through defining fewer points to represent the boundary of the obstacle but that compromises the quality of the method.

4.3.3 Use case C

In this use case, the whole system is implemented from planning to guidance and control. The small red dots, represent the global planning created Asymmetrical BI-RRT, presented in the form of WPs, used to generate the path which is then followed by the robot.

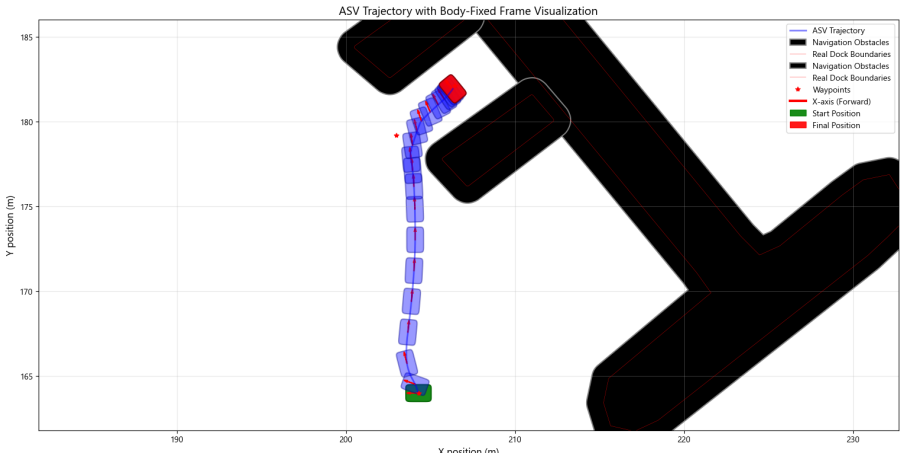
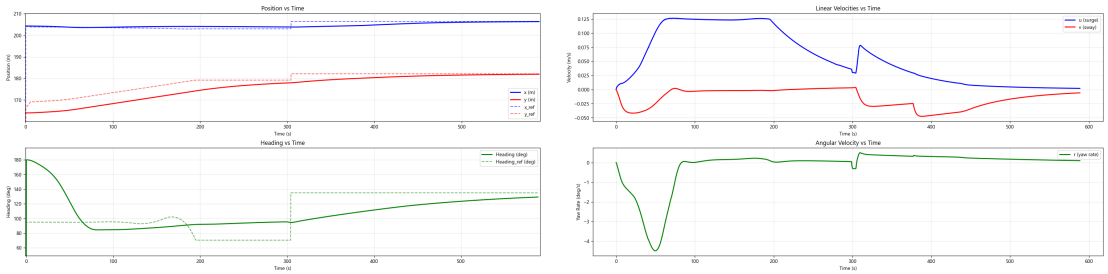
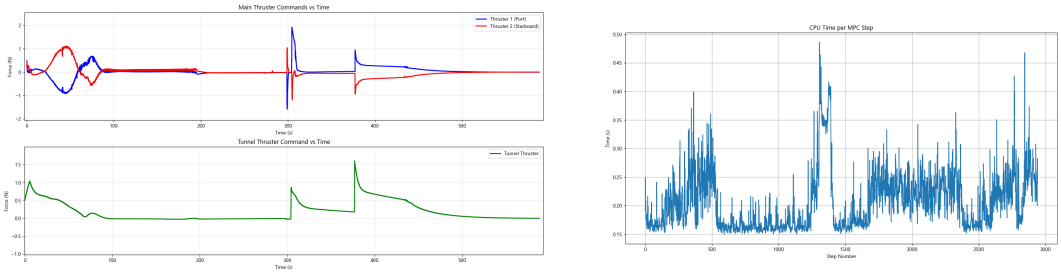


Figure 4.13: Use Case C (heading not aligned) : Short movement for the dock, simulation



(a) (b)

Figure 4.14: Use Case C (heading not align) : Linear and angular velocities and $[x, y, \psi]$ changing over time



(a) (b)

Figure 4.15: Use Case C (heading not align): Thrusters input variation along time and time to solve the OCP problem

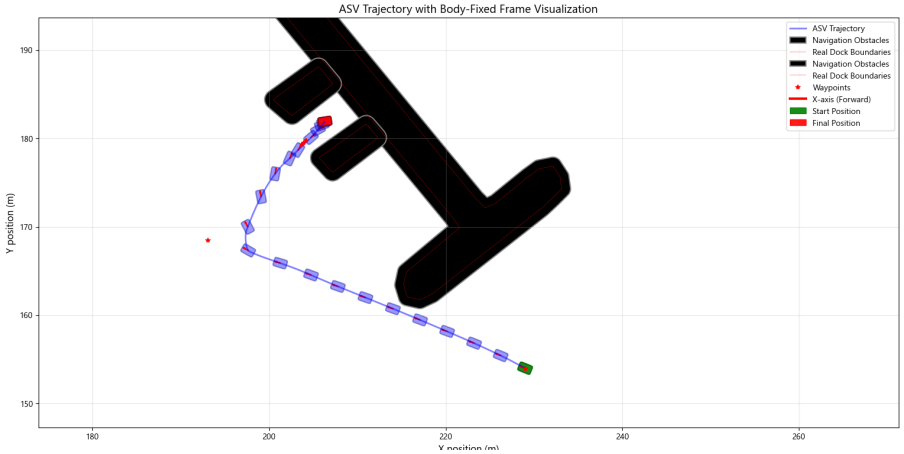
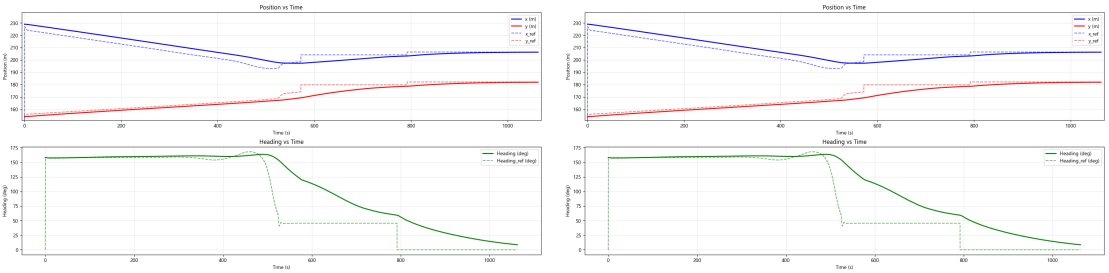


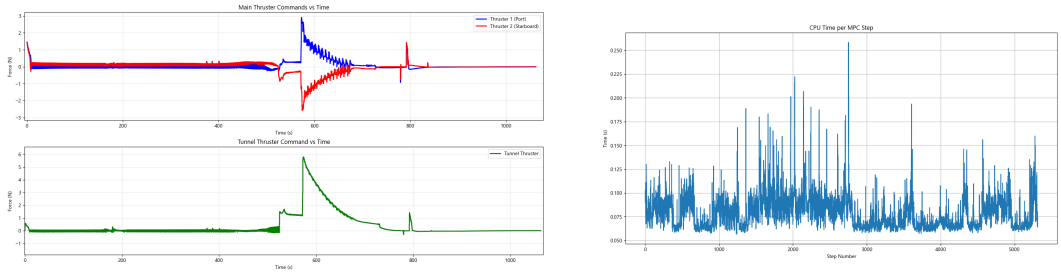
Figure 4.16: Use Case C (heading aligned) : Larger movement simulation

In the two images 4.13 and 4.16 it is possible the see that the vehicle had to deviate his path from the obstacle even though passing very close to it, which is the advantage of the hard constraints.



(a) (b)

Figure 4.17: Use Case C (heading aligned) : Linear and angular velocities and $[x, y, \psi]$ changing over time



(a) (b)

Figure 4.18: Use Case C (heading aligned) : Thrusters input variation along time and time to solve the OCP problem

Graphs with the speed over time, show that the robot is able to reach the desired targets and, in the end, stop in the final desired pose.

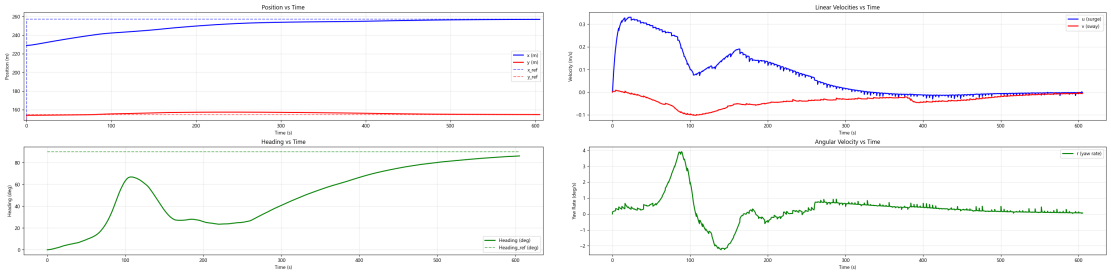
4.3.4 Introducing Noise

To try and test how the control would do under harsher and more realistic circumstances, use cases PH and C were tested in the same way as before but with the an ciclic error on thruster's input which is simulated by adding noise to the thruster value every 20 iterations following an Normal distribution between $[-2, 2]N$.

Use Case PH

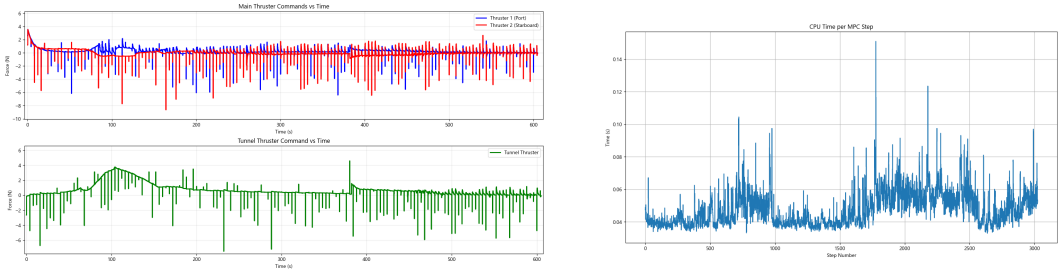


Figure 4.19: Use Case PH : Simulation with noise on thruster's input.



(a) (b)

Figure 4.20: Use Case Use Case PH with noise : Linear and angular velocities and $[x, y, \psi]$ changing over time



(a) (b)

Figure 4.21: Use Case Use Case PH with noise : Thrusters input variation along time and time to solve the OCP problem

In figure 4.19 and in the following graphs, it's possible to see the the controller is still able to guide the boat to the desired position and stop in the desired pose.

Use Case C

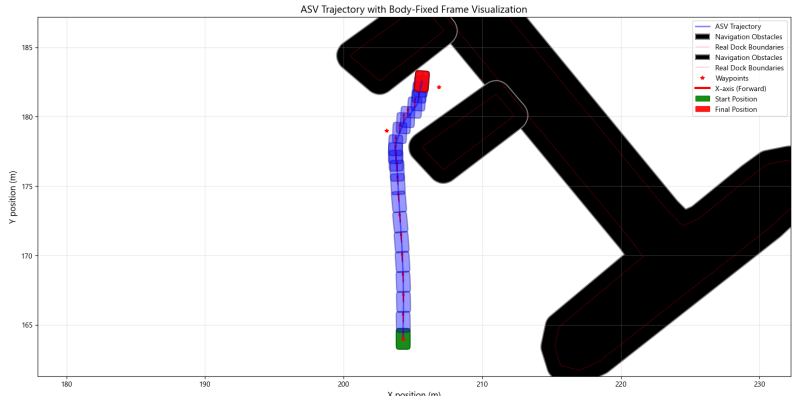
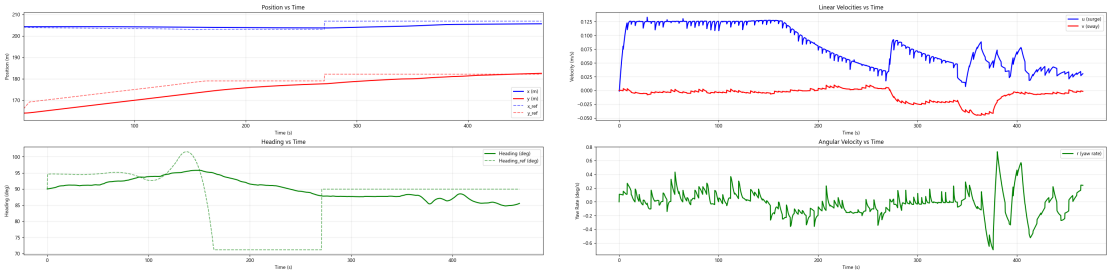
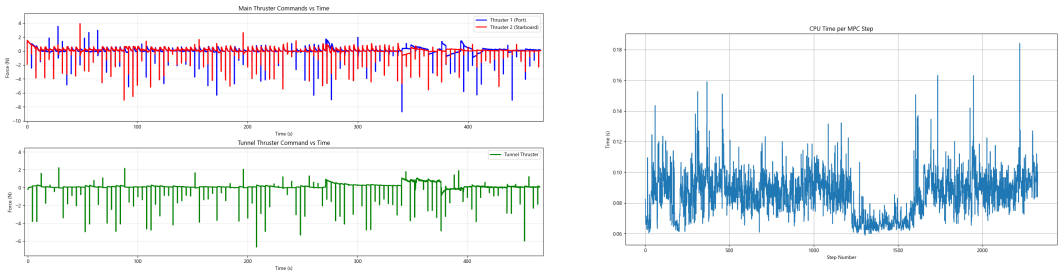


Figure 4.22: Use Case C : Simulation with noise on thruster's input.



(a) (b)

Figure 4.23: Use Case Use Case C with noise : Linear and angular velocities and $[x, y, \psi]$ changing over time



(a) (b)

Figure 4.24: Use Case Use Case C with noise : Thrusters input variation along time and time to solve the OCP problem

For use case C 4.22 the noise made the vehicle hit the docking during the final movements but MPC still showed stability capabilities under the tested harsh environments.

4.4 Discussion

In terms of speed, the vehicle goes considerably slow, and that is a weak point of this work. The vehicle only goes faster when it goes directly to a far WP, which only happens between the last 2 WPs. This could be changed with a finer tuning of the penalization parameters, possibly forcing a higher speed in the vehicle's x body frame axis.

The vehicle shows stability throughout the whole process as seen in the images and graphs. Even so, with the introduction of obstacle hard constraints, int some situations the controller wasn't able to solve the OCP, which became infeasible, creating less controlled situations.

The obstacle avoidance method is the weakest point of this dissertation although it brings a new perspective into static obstacle avoidance and how this could be introduced in control. Since it only uses distances, it could be even integrated with sensor readings in the future. The method has a clear effect in the control but it is not yet generalizable due to mainly computational efficiency problems. For further implementations or real world testes, it would need to be tested more intensively.

Time taken on each iteration was showed but these are highly dependent on the machine where these were ran, so these values are highly volatile and it's meaning is less relevant than the rest

Further simulations should be done using realistic wave effects or different noise considerations but, for the essential process of docking, the MPC is able to guide the vehicle towards the target position and stabilize it with the desired heading, concluding the initial objectives of this work.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This paper presented a comprehensive Sequential Linearization Model Predictive Control framework for autonomous surface vehicle navigation. The methodology successfully addresses the computational challenges associated with nonlinear MPC while maintaining adequate control performance for trajectory tracking and obstacle avoidance.

In this dissertation, we set out to develop and validate a robust Model Predictive Control framework for ASV docking operations. The primary objective was to achieve precise trajectory tracking, which was succesfully achieved. And then were also suggested to introduce efficient obstacle avoidance which was achieved in controlled conditions.

To accomplish these goals, we implemented a comprehensive MPC-based guidance and control architecture, integrating a dynamic Line-of-Sight (LOS) guidance strategy and constraints for static obstacle avoidance. The framework was tested and validated through simulation scenarios, revealing relevant results in docking precision.

The key findings from this research include:

- Real-time trajectory tracking with maneuver precision for homing.
- Successful integration of static obstacle avoidance in OCP as hard constraints.
- A robust control architecture for ASV docking, using an MPC-based architecture suitable for complex scenarios.
- An open foundation for future developments in maritime autonomous navigation and control.

Despite these advancements, certain limitations were found. The computational demand of the MPC solver currently restricts its implementation on hardware with limited computational resources especially when adding obstacle contraints. Additionally, experimental validation was confined to simulation scenarios, which may not fully replicate unpredictable real-world conditions.

Overall, this dissertation demonstrates the viability and effectiveness of MPC-based control strategies for autonomous maritime operations, establishing a solid foundation for future research.

5.2 Future Work

This project intends to be part of a small step for the development of marine robotics, as it is a very complex and rich field. This project opens the door for MPC applications in docking scenarios, specially through the possibility of introducing different types of geometric obstacles in the cost to ensure safety. In future work, it's necessary to introduce environmental disturbances, as this need to be tested, before real-life tests. A continuity of this work would go around methods for introducing obstacle constraints in OCP, in a more clear and tested version. It would probably also dive into sensing methodologies on how to introduce sensors for real-time obstacle avoidance. This stands for static obstacles but also for dynamic obstacles, as it is a very important scenario which this work does not approach. It would be interesting to see how the introduction of AI methodologies could help, as the introduction of Neural Network to substitute the nonlinear dynamics equation is already suggested but yet to be intensively tested. Last but not least, real-world validation would also be an important contribution for the statement of MPC in marine scenarios.

References

[1] J.H. Ahlberg, E.N. Nilson, and J.L. Walsh. *The Theory of Splines and Their Applications*. Academic Press, 1967.

[2] Kristoffer Bergman et al. *An Optimization-Based Motion Planner for Autonomous Maneuvering of Marine Vessels in Complex Environments*. May 2020. DOI: 10.48550/arXiv.2005.02674.

[3] Pierre Bézier. "How Renault Uses Numerical Control for Car Body Design and Tooling". In: *SAE International Congress* 760 (1968), pp. 1–13. DOI: 10.4271/680010.

[4] Rakesh P. Borase et al. "A review of PID control, tuning methods and applications". In: *International Journal of Dynamics and Control* 9.2 (2021), pp. 818–827. ISSN: 2195-2698. DOI: 10.1007/s40435-020-00665-4. URL: https://doi.org/10.1007/s40435-020-00665-4.

[5] Virginia Burkett. "Global climate change implications for coastal and offshore oil and gas development". In: *Energy policy* 39.12 (2011), pp. 7719–7725.

[6] L. Caiza et al. "Reducing energy consumption in an industrial process by using model predictive control". In: *2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)* (2017). DOI: 10.1109/ropec.2017.8261618.

[7] Daniel Filipe Campos, Aníbal Matos, and Andry Maykol Pinto. "Multi-domain inspection of offshore wind farms using an autonomous surface vehicle". In: *SN Applied Sciences* 3.4 (2021). Cited by: 27; All Open Access, Gold Open Access. DOI: 10.1007/s42452-021-04451-5. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85102521310&doi=10.1007%2fs42452-021-04451-5&partnerID=40&md5=bf70ddb37f314e15dfbf935519e68996.

[8] J. Chavez-Galaviz et al. "ASV station keeping under wind disturbances using neural network simulation error minimization model predictive control". In: *Journal of Field Robotics* 41.6 (2024), pp. 1797–1813. DOI: 10.1002/rob.22346. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85191288434&doi=10.1002%2frob.22346&partnerID=40&md5=646d86597ec0c4cabe7722e2be6c3fd4.

[9] Qi Chen, Yanan Qin, and Gelun Li. "QPSO-MPC based tracking algorithm for cable-driven continuum robots". In: *Frontiers in Neurorobotics* 16 (2022). ISSN: 1662-5218. DOI: 10.3389/fnbot.2022.1014163. URL: https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2022.1014163.

[10] Li Dai et al. "Robust Model Predictive Tracking Control for Robot Manipulators With Disturbances". In: *IEEE Transactions on Industrial Electronics* 68 (2021), pp. 4288–4297. DOI: 10.1109/TIE.2020.2984986.

[11] Edsger W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1.1 (1959), pp. 269–271.

[12] Jinxin Dong et al. “Climate Change Impacts on Coastal and Offshore Petroleum Infrastructure and the Associated Oil Spill Risk: A Review”. In: *Journal of Marine Science and Engineering* 10.7 (2022). ISSN: 2077-1312. DOI: 10.3390/jmse10070849. URL: https://www.mdpi.com/2077-1312/10/7/849.

[13] F. Fahimi. “Sliding mode formation control for under-actuated autonomous surface vehicles”. In: *2006 American Control Conference* (2006), 6 pp.-. DOI: 10.1109/ACC.2006.1657387.

[14] Tingxiang Fan et al. “Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios”. In: *The International Journal of Robotics Research* 39.7 (2020), pp. 856–892. DOI: 10.1177/0278364920916531.

[15] Paulo Flores and Hamid Lankarani. *Formulation for Planar Multibody Dynamics*. Jan. 2012.

[16] Thor Fossen. *Guidance and Control of Ocean Vehicles*. 1st. Chichester, UK: Wiley, 1994.

[17] Thor I Fossen, Kristin Y Pettersen, and Roberto Galeazzi. “Line-of-sight path following for dubins paths with adaptive sideslip compensation of drift forces”. In: *IEEE transactions on control systems technology* 23.2 (2014), pp. 820–827.

[18] Marguerite Frank and Philip Wolfe. “An algorithm for quadratic programming”. In: *Naval Research Logistics Quarterly* 3.1-2 (1956), pp. 95–110. DOI: 10.1002/nav.3800030109.

[19] H. Gao, Z. Kan, and K. Li. “Robust lateral trajectory following control of unmanned vehicle based on model predictive control”. In: *IEEE/ASME Transactions on Mechatronics* 27 (3 2021), pp. 1278–1287. DOI: 10.1109/tmech.2021.3087605.

[20] Yong Gao et al. “Trajectory tracking of multi-legged robot based on model predictive and sliding mode control”. In: *Information Sciences* 606 (2022), pp. 489–511. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2022.05.069. URL: https://www.sciencedirect.com/science/article/pii/S0020025522005011.

[21] B. Geranmehr and K. Vafaee. “Hybrid adaptive neural network auv controller design with sliding mode robust term”. In: *International Journal of Maritime Technology* 7 (2017), pp. 49–55. DOI: 10.18869/acadpub.ijmt.7.49.

[22] Guoqiang Hao et al. “UAV Path Planning Based on Improved Artificial Potential Field Method”. In: *Aerospace* 10.6 (2023). ISSN: 2226-4310. DOI: 10.3390/aerospace10060562. URL: https://www.mdpi.com/2226-4310/10/6/562.

[23] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[24] Junwoo Jang and Jinwhan Kim. “Hydrodynamic modeling of a robotic surface vehicle using representation learning for long-term prediction”. In: *Ocean Engineering* 270 (2023), p. 113620. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2023.113620. URL: https://www.sciencedirect.com/science/article/pii/S0029801823000045.

[25] Tao Jiang, Yan Yan, and Shuang-He Yu. “Adaptive Sliding Mode Control for Unmanned Surface Vehicles with Predefined-Time Tracking Performances”. In: *Journal of Marine Science and Engineering* 11.6 (2023). ISSN: 2077-1312. DOI: 10.3390/jmse11061244. URL: https://www.mdpi.com/2077-1312/11/6/1244.

[26] Xiangkui Jiang, Zihao Wang, and Chao Dong. "A Path Planning Algorithm Based on Improved RRT Sampling Region". In: *Computers, Materials Continua* 80 (Sept. 2024), pp. 4303–4323. DOI: 10.32604/cmc.2024.054640.

[27] Peisen Jin et al. "Construction of Simulation System for USV Motion Control and Design of Multi-Mode Controllers Based on VRX and Simulink". In: *Applied Sciences* 15 (Apr. 2025), p. 4213. DOI: 10.3390/app15084213.

[28] 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.

[29] Lydia E Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[30] Y. Kebbati et al. "Optimized self-adaptive pid speed control for autonomous vehicles". In: *2021 26th International Conference on Automation and Computing (ICAC)* (2021), pp. 1–6. DOI: 10.23919/icac50006.2021.9594131.

[31] Alex Kendall et al. "Learning to Drive in a Day". In: *International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8248–8254. DOI: 10.1109/ICRA.2019.8793742.

[32] Oussama Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. 1986, pp. 500–505.

[33] M. Kim et al. "Development of local path planning using selective model predictive control, potential fields, and particle swarm optimization". In: *Robotics* 13 (3 2024), p. 46. DOI: 10.3390/robotics13030046.

[34] Sven Koenig and Maxim Likhachev. "Fast replanning for navigation in unknown terrain". In: *IEEE Transactions on Robotics* 21.3 (2005), pp. 354–363.

[35] Steven M. LaValle. "Rapidly-exploring random trees: A new tool for path planning". In: *Technical Report, Computer Science Dept., Iowa State University* (2001).

[36] Anastasios M. Lekkas and Thor I. Fossen. "Minimization of cross-track and along-track errors for path tracking of marine underactuated vehicles". In: *2014 European Control Conference (ECC)*. 2014, pp. 3004–3010. DOI: 10.1109/ECC.2014.6862594.

[37] Simon J.N. Lexau, Anastasios M. Lekkas, and Morten Breivik. "Nonlinear PID Control for Automatic Docking of a Large Container Ship in Confined Waters Under the Influence of Wind and Currents∗∗This work was supported by the Research Council of Norway under Project 309230 (SFI AutoShip)." In: *IFAC-PapersOnLine* 58.20 (2024). 15th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles CAMS 2024, pp. 265–272. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2024.10.065. URL: https://www.sciencedirect.com/science/article/pii/S2405896324018202.

[38] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2016). URL: https://arxiv.org/abs/1509.02971.

[39] Aleksey Logunov et al. "Polygon decomposition for obstacle representation in motion planning with Model Predictive Control". In: *Engineering Applications of Artificial Intelligence* 153 (2025), p. 110690. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2025.110690. URL: https://www.sciencedirect.com/science/article/pii/S0952197625006906.

[40] Jakob Løver, Vilde B. Gjærum, and Anastasios M. Lekkas. "Explainable AI methods on a deep reinforcement learning agent for automatic docking∗∗This work was supported by the Research Council of Norway through the EXAIGON project, project number 304843." In: *IFAC-PapersOnLine* 54.16 (2021). 13th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2021, pp. 146–152. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.10.086. URL: https://www.sciencedirect.com/science/article/pii/S2405896321014889.

[41] I. Malouche and F. Bouani. "Realtime application of constrained predictive control for mobile robot navigation". In: *International Journal of Advanced Computer Science and Applications* 8 (11 2017). DOI: 10.14569/ijacsa.2017.081136.

[42] Andreas B. Martinsen, Anastasios M. Lekkas, and Sebastien Gros. "Autonomous docking using direct optimal control". In: *IFAC-PapersOnLine* 52.21 (2019). 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2019, pp. 97–102. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.12.290. URL: https://www.sciencedirect.com/science/article/pii/S2405896319321755.

[43] Andreas Bell Martinsen et al. "Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments". In: *IEEE Access* 8 (Nov. 2020), pp. 204974–204986. DOI: 10.1109/ACCESS.2020.3037171.

[44] Mao-Wei Nilsson. "Autonomous Docking and Navigation of ships using Model Predictive Control". Master's thesis. Stockholm, Sweden: KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science, 2018.

[45] Petter K. Ødven, Andreas B. Martinsen, and Anastasios M. Lekkas. "Static and dynamic multi-obstacle avoidance and docking of ASVs using computational geometry and numerical optimal control". In: *IFAC-PapersOnLine* 55.31 (2022). 14th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2022, pp. 50–57. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2022.10.408. URL: https://www.sciencedirect.com/science/article/pii/S2405896322024545.

[46] B. R. Page et al. "Underwater docking approach and homing to enable persistent operation". In: *Frontiers in Robotics and AI* 8 (2021). DOI: 10.3389/frobt.2021.621755.

[47] Jan-David Quesel et al. "How to model and prove hybrid systems with KeYmaera: a tutorial on safety". In: *International Journal on Software Tools for Technology Transfer* 18 (Feb. 2016), pp. 67–91. DOI: 10.1007/s10009-015-0367-0.

[48] E. Rimon and D.E. Koditschek. "Exact robot navigation using artificial potential functions". In: *IEEE Transactions on Robotics and Automation* 8.5 (1992), pp. 501–518. DOI: 10.1109/70.163777.

[49] Isaac J. Schoenberg. "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions". In: *Quarterly of Applied Mathematics* 4 (1946), pp. 45–99. DOI: 10.1090/qam/15914.

[50] Enzhong Shan et al. "A Dynamic RRT Path Planning Algorithm Based on B-Spline". In: *2009 Second International Symposium on Computational Intelligence and Design*. Vol. 2. 2009, pp. 25–29. DOI: 10.1109/ISCID.2009.155.

[51] M. F. d. Silva et al. “Project and control allocation of a 3 dof autonomous surface vessel with aerial azimuth propulsion system”. In: *IEEE Access* 9 (2021), pp. 5212–5227. DOI: 10.1109/access.2020.3048330.

[52] Anthony Stentz. “Optimal and Efficient Path Planning for Partially-Known Environments”. In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. 1994, pp. 3310–3317.

[53] Ting Sun, Cheng Liu, and Yong Yin. “Optimized SARLOS guidance for berthing control of autonomous surface vehicle with collision avoidance”. In: *Ocean Engineering* 318 (2025), p. 120115. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2024.120115. URL: https://www.sciencedirect.com/science/article/pii/S002980182403453X.

[54] Wanxing Tang et al. “Dual-Arm Robot Trajectory Planning Based on Deep Reinforcement Learning under Complex Environment”. In: *Micromachines* 13.4 (2022). ISSN: 2072-666X. DOI: 10.3390/mi13040564. URL: https://www.mdpi.com/2072-666X/13/4/564.

[55] X. Tang et al. “Weight adaptive path tracking control for autonomous vehicles based on pso-bp neural network”. In: *Sensors* 23 (1 2022), p. 412. DOI: 10.3390/s23010412.

[56] Ken B. H. Teo, Edgar An, and Pierre Beaujean. “A Robust Fuzzy Autonomous Underwater Vehicle (AUV) Docking Approach for Unknown Current Disturbances”. In: *IEEE Journal of Oceanic Engineering* 37 (2012), pp. 143–155. DOI: 10.1109/JOE.2011.2180058.

[57] T. G. Thuruthel et al. “Stable Open Loop Control of Soft Robotic Manipulators”. In: *IEEE Robotics and Automation Letters* 3 (2018), pp. 1292–1298. DOI: 10.1109/LRA.2018.2797241.

[58] Thomas George Thuruthel et al. “Learning Closed Loop Kinematic Controllers for Continuum Manipulators in Unstructured Environments.” In: *Soft robotics* 4 3 (2017), pp. 285–296. DOI: 10.1089/soro.2016.0051.

[59] Antonio Vasilijević et al. “Coordinated navigation of surface and underwater marine robotic vehicles for ocean sampling and environmental monitoring”. In: *IEEE/ASME Transactions on Mechatronics* 22.3 (2017). Cited by: 156, pp. 1174–1184. DOI: 10.1109/TMECH.2017.2684423. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85026788790&doi=10.1109%2fTMECH.2017.2684423&partnerID=40&md5=05e6f76b37046a824a74726046f26b6e.

[60] Andreas Wächter and Lorenz T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical Programming* 106.1 (2006), pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

[61] F. Wang, Z. Hu, and V. Veerasamy. “Pso-based model predictive control for load frequency regulation with wind turbines”. In: *Energies* 15 (21 2022), p. 8219. DOI: 10.3390/en15218219.

[62] Dustin J. Webb and Jur van den Berg. “Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 5054–5061. DOI: 10.1109/ICRA.2013.6631299.

[63] Zhenping Wu et al. “Fast-RRT: A RRT-Based Optimal Path Finding Method”. In: *Applied Sciences* 11.24 (2021). ISSN: 2076-3417. DOI: 10.3390/app112411777. URL: https://www.mdpi.com/2076-3417/11/24/11777.

[64] Peng Xin et al. “Improved Bidirectional RRT* Algorithm for Robot Path Planning”. In: *Sensors* 23.2 (2023). ISSN: 1424-8220. DOI: 10.3390/s23021041. URL: https://www.mdpi.com/1424-8220/23/2/1041.

[65] Altay Zhakatayev, Matteo Rubagotti, and H. A. Varol. “Closed-Loop Control of Variable Stiffness Actuated Robots via Nonlinear Model Predictive Control”. In: *IEEE Access* 3 (2015), pp. 235–248. DOI: 10.1109/ACCESS.2015.2418157.

[66] Xin Zhang, Dexuan Zou, and Xin Shen. “A Novel Simple Particle Swarm Optimization Algorithm for Global Optimization”. In: *Mathematics* 6.12 (2018). ISSN: 2227-7390. DOI: 10.3390/math6120287. URL: https://www.mdpi.com/2227-7390/6/12/287.

[67] Yongchang Zhang et al. “3D path following control for UAVs using L1 adaptive method”. In: *2015 Chinese Automation Congress (CAC)*. 2015, pp. 1098–1104. DOI: 10.1109/CAC.2015.7382662.

[68] Yongding Zhang et al. “MPC-based 3-D trajectory tracking for an autonomous underwater vehicle with constraints in complex ocean environments”. In: *Ocean Engineering* (2019). DOI: 10.1016/J.OCEANENG.2019.106309.

[69] P. Zhao et al. “Design of a control system for an autonomous vehicle based on adaptive-pid”. In: *International Journal of Advanced Robotic Systems* 9 (2 2012). DOI: 10.5772/51314.

[70] Huarong Zheng, Rudy R. Negenborn, and Gabriel Lodewijks. “Trajectory tracking of autonomous vessels using model predictive control”. In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 8812–8818. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20140824-6-ZA-1003.00767. URL: https://www.sciencedirect.com/science/article/pii/S1474667016430041.

[71] Jian Zheng, Jiayin Hu, and Yun Li. “Codesign of dynamic collision avoidance and trajectory tracking for autonomous surface vessels with nonlinear model predictive control”. In: *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment* 236.4 (2022), pp. 938–952. DOI: 10.1177/14750902221088111. eprint: https://doi.org/10.1177/14750902221088111. URL: https://doi.org/10.1177/14750902221088111.

Appendix A

Complete Dynamic Model Assembly

Rigid-Body Inertia Matrix with Values:

$$M_{RB} = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 7.73 \end{bmatrix} \text{ kg, kg·m}^2 \tag{A.1}$$

Added Mass Matrix with Values:

$$M_A = \begin{bmatrix} 5.22 & 0 & 0 \\ 0 & 46.39 & 1.98 \\ 0 & 1.98 & 48.24 \end{bmatrix} \text{ kg, kg·m, kg·m}^2 \tag{A.2}$$

Total System Inertia Matrix:

$$M = M_{RB} + M_A = \begin{bmatrix} 55.22 & 0 & 0 \\ 0 & 96.39 & 1.98 \\ 0 & 1.98 & 55.97 \end{bmatrix} \text{ kg, kg·m, kg·m}^2 \tag{A.3}$$

Rigid-Body Coriolis Matrix with Values:

$$C_{RB}(\nu) = \begin{bmatrix} 0 & 0 & -50v \\ 0 & 0 & 50u \\ 50v & -50u & 0 \end{bmatrix} \text{ N} \tag{A.4}$$

Hydrodynamic Coriolis Matrix with Values:

$$C_A(\nu) = \begin{bmatrix} 0 & 0 & 46.39v + 1.98r \\ 0 & 0 & -5.22u \\ -46.39v - 1.98r & 5.22u & 0 \end{bmatrix} \text{ N} \tag{A.5}$$

Total Coriolis Matrix:

$$C(\nu) = C_{RB}(\nu) + C_A(\nu) = \begin{bmatrix} 0 & 0 & -96.39v - 1.98r \\ 0 & 0 & 55.22u \\ 96.39v + 1.98r & -55.22u & 0 \end{bmatrix} \text{ N} \tag{A.6}$$

Damping Matrix with Values:

$$D(\nu) = \begin{bmatrix} -8.27|u| & 0 & 0 \\ 0 & -338.21|v| & -8.96|r| \\ 0 & -8.96|v| & -96.17|r| \end{bmatrix} \text{ N, N·m} \tag{A.7}$$

Control Allocation Matrix with Values:
With the geometric configuration parameters:

- $r_1 = 0.335$ m: Half the distance between the two bow thrusters
- $r_3 = 0.3$ m: Distance from the tunnel thruster to the vessel's center of rotation

$$T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & -1 \\ 0.335 & -0.335 & -0.3 \end{bmatrix} \text{ N/N, m}^{-1} \tag{A.8}$$

Complete model for the Dynamics:
Combining all components, the complete ASV dynamics become:

$$\begin{bmatrix} 55.22 & 0 & 0 \\ 0 & 96.39 & 1.98 \\ 0 & 1.98 & 55.97 \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 & 0 & -96.39v - 1.98r \\ 0 & 0 & 55.22u \\ 96.39v + 1.98r & -55.22u & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{A.9}$$

$$+ \begin{bmatrix} -8.27|u| & 0 & 0 \\ 0 & -338.21|v| & -8.96|r| \\ 0 & -8.96|v| & -96.17|r| \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & -1 \\ 0.335 & -0.335 & -0.3 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_{tunnel} \end{bmatrix} \tag{A.10}$$