Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Henrik Rohde Nordhus

COLREGs-aware automatic docking for autonomous surface vessels

Master's thesis in Cybernetics and Robotics
Supervisor: Anastasios M. Lekkas
Co-supervisor: Morten Breivik, Thomas Johansen
June 2023

NTNU
Norwegian University of
Science and Technology

Henrik Rohde Nordhus

COLREGs-aware automatic docking for autonomous surface vessels

Master's thesis in Cybernetics and Robotics
Supervisor: Anastasios M. Lekkas
Co-supervisor: Morten Breivik, Thomas Johansen
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

In an era where autonomous systems are increasingly shaping industries and everyday life, the field of maritime navigation has witnessed a significant transformation. This Master's thesis showcases the advancements made in docking procedures, specifically for Autonomous Surface Vessels (ASVs), within the dynamic and intricate maritime environment. It presents an exploration into the complex problem of docking ASVs while adhering to the Convention on the International Regulations for Preventing Collisions at Sea (COLREGs).

Recent work by Ødven et al. (2022) has shown promise in solving the docking problem for ASVs by combining computational geometry and nonlinear control theory. Triangulation is utilized to generate waypoints at safe distances from harbors and obstacles. A search algorithm navigates through these waypoints to identify the shortest collision-free path from the initial location to the docking point. Subsequently, a nonlinear optimal control problem (NOCP) is numerically solved to yield the optimal trajectory and control sequence corresponding to this path. Spatial constraints representing the harbor and any obstacles are seamlessly integrated into this approach to ensure safe navigation.

In an innovative extension of these methods, this thesis utilizes a collision avoidance (COLAV) system considering the COLREGs to identify rules relevant to a specific scenario, thereby facilitating realistic ASV maneuvering. Other contributions include a path-smoothing algorithm that ensures a collision-free path and takes COLREGs into account and an adaptive approach to adjusting the number of time steps in the MPC to balance computational time and accuracy.

The findings in this thesis contribute to the growing field of ASV research, offering innovative strategies and algorithms that enhance docking safety, efficiency, and compliance with international regulations. Moreover, the results show that combining triangulation and search algorithms to obtain a collision-free path, combined with optimal control theory, presents a feasible and promising solution for automating the docking problem. By employing an optimization-based approach, the vessel can uphold its dynamics while accounting for the constraints of actuators and spatial limitations.

Preface

This thesis is a culmination of my Master's degree at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU). It has been a journey through the trials and tribulations, triumphs and breakthroughs, of seeking a solution to the docking problem that autonomous surface vessels (ASVs) face. My hope is that the findings presented here will open doors for future research and development in the area of ship autonomy and advance the field toward safer, more efficient, and reliable automatic docking procedures.

Several initiatives are currently engaged in researching and conducting experiments on autonomous ships. Notably, recent advancements utilizing optimal control theory have shown promising results regarding the automatic docking of ASVs. One such approach, proposed by Ødven et al. (2022), draws inspiration from the works of Martinsen et al. (2019) and Bitar et al. (2019). Building upon the foundation laid by these prior studies, this Master's thesis serves as a continuation of their research, as well as an extension of my specialization project (Nordhus (2022)). It will, therefore draw upon many similar aspects regarding motivation, theory, and methodology. The relevant information has therefore been included in this thesis, with some adjustments. The abstract, preface, and introduction are inspired by and draw on resembling content to the project thesis. The complete list of included contents are: Chapter 2 - Section 2.1, 2.2.1, 2.2.3, and 2.4; and Chapter 3 - Section 3.1.1, 3.1.3, 3.2, and 3.3.

This work illustrates how a complex task can be divided into manageable segments, combining computational geometry, search algorithms, optimal control theory, and collision avoidance systems while considering COLREGs. To manage such a task, the versatile, flexible, and powerful language, *Python*, was utilized for the implementation of these methods and algorithms. *CasADi*, developed by Andersson et al. (2018), offers an efficient approach for implementing and computing solutions to the nonlinear optimal control problem. All of the work in this thesis has been developed from scratch with helping advice from supervisors and fellow students. The simulation and computation work was carried out on a Dell Optiplex 7090.

I would like to thank my supervisors Anastasios Lekkas, Morten Breivik, and Thomas Johansen, for the valuable discussions, helpful feedback, and insights regarding the docking problem. Furthermore, I extend my thanks to Simon Lexau for offering assistance whenever I needed help. I am truly grateful to my fellow students, whose presence has made each day of this semester exciting and memorable. Lastly, heartfelt thanks go to my partner and my family, whose enduring support and positivity have been invaluable in this endeavor.

13.06.2023

Sammendrag

I en æra hvor autonome systemer i økende grad former industrier og hverdagsliv, har området for maritim navigasjon opplevd en betydelig transformasjon. Denne masteroppgaven viser fremskrittene som er gjort i dokkingsprosedyrer, spesielt for autonome overflatefartøy (ASV), innenfor dynamiske og komplekse maritime områder. Den presenterer det komplekse problemet med å dokke ASV-er samtidig som man overholder Konvensjonen om internasjonale regler for å forhindre kollisjoner til sjøs (COLREGs).

Nylig arbeid av Ødven et al. (2022) har vist lovende resultater i forbindelse med dokking for autonome overflatefartøy (ASV) ved å kombinere geometri og ikke-lineær kontrollteori. Triangulering brukes til å generere veipunkter med tryg avstand fra havner og hindringer. En søkealgoritme navigerer gjennom disse veipunktene for å identifisere den korteste kollisjonsfrie banen fra den opprinnelige plasseringen til fortøyningspunktet. Deretter løses et ikke-lineært optimalt kontrollproblem (NOCP) numerisk for å gi den optimale kontrollsekvensen som passer til denne banen. Begrensninger som representerer havnen og eventuelle hindringer er sømløst integrert i denne tilnærmingen for å sikre sikker navigasjon.

I en innovativ utvidelse av disse metodene, benytter denne avhandlingen et kollisjonsforebyggende (COLAV) system som tar hensyn til COLREGs for å identifisere hvilke regler som er relevante for et spesifikt scenario, og dermed legge til rette for realistiske ASV-manøvrer. Andre bidrag inkluderer en ny baneglattende algoritme som sikrer en kollisjonsfri bane og tar hensyn til COLREGs, samt en adaptiv tilnærming til å justere antall tidssteg i MPC for å balansere beregningstid og nøyaktighet.

Funnene i denne avhandlingen bidrar til det voksende feltet av ASV-forskning, og tilbyr innovative strategier og algoritmer som forbedrer dokkingsikkerhet, effektivitet og overholdelse av internasjonale forskrifter. Videre viser resultatene at kombinasjonen av triangulering og søkealgoritmer for å oppnå en kollisjonsfri bane, kombinert med optimal kontrollteori, presenterer en gjennomførbar og lovende løsning for automatisering av dokkingproblemet. Ved å bruke en optimaliseringsbasert tilnærming, kan fartøyet opprettholde sin dynamikk mens det tar hensyn til begrensningene av aktuatorer og andre fysiske begrensninger.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Abbreviations

Abbreviation	Description
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARPA	Automatic Radar Plotting Aid
ASV	Autonomous Surface Vessel
BODY	Body-Fixed
CB	Constant Bearing
COLAV	Collision Avoidance
COLREGs	Convention on the International Regulations for Preventing Collisions at Sea
DDPG	Deep Deterministic Policy Gradient
DOF	Degree of Freedom
DP	Dynamic Positioning
DRL	Deep Reinforcement Learning
ECI	Earth-Centered-Inertial
GW	Give-way
HO	Head-on
IPOPT	Interior Point Optimizer
LOS	Line-of-Sight
ML	Machine Learning
MPC	Model Predictive Control
NED	North-East-Down
NLMPC	Nonlinear Model Predictive Control
NLP	Nonlinear Programming
NOCP	Nonlinear Optimal Control Problem
NP	Nondeterministic Polynomial time
OCP	Optimal Control Problem
OS	Ownship
OT	Overtaking
PID	Proportional–Integral–Derivative
PP	Pure-Pursuit
PPO	Proximal Policy Optimization
PRM	Probabilistic Roadmap
RL	Reinforcement Learning
RL-MPC	Reinforcement Learning based Model Predictive Control
ROC	Region of Collision
RRT	Rapidly Exploring Random Tree
SF	Safe
SNAME	Society of Naval Architects and Marine Engineers
SO	Stand-on

TS	Target Ship

1

Introduction

1.1 Background and motivation

The evolving research domain of *autonomous systems* has gained significant traction over the past few decades. The drive towards autonomy arises from goals such as enhancing efficiency and safety, mitigating costs and emissions, and accomplishing tasks deemed monotonous for humans. Many sectors within maritime operations, including transportation, seabed mapping, underwater inspection, and service, could benefit from increased autonomy. Currently, shipping accounts for 90% of worldwide freight transport [12]. A study by Allianz reveals that 75% of marine accidents are due to human error, presumably related to evasive maneuvers to prevent collisions [13]. Recent progress in autonomous vehicles shows the potential for achieving high situational awareness and autonomy. Therefore, autonomous surface vessels (ASVs) are considered pivotal for constructing a safe and sustainable shipping industry.

Various initiatives have explored automating the shipping industry, particularly in areas such as path planning and autonomous docking. The Centers for Research-based Innovation (SFI), funded by The Research Council of Norway, is one of these initiatives. SFI's primary objective is to augment the commercial sector's capacity to innovate and generate value through a focus on long-term research. One crucial project within SFI is the SFI AutoShip, established eight years ago to reinforce Norway's position in safe autonomous ships for sustainable operations [14, 15].

Several companies have conducted experiments and developed full-scale ASVs. For instance, Rolls Royce (now Kongsberg Maritime) and Finferries developed Falco, the world's first autonomous ferry, in 2018. This ferry can follow a predefined course, modify it to avoid collisions, adjust its speed, and dock autonomously [16]. Kongsberg Maritime and Bastø Fosen collaborated to conduct the world's first adaptive ferry transit in 2020 [17]. Other innovative developments in autonomous shipping have emerged in the Norwegian maritime area in recent years. Varying from research projects investigating sensor fusion

Figure 1.1: Picture from the opening ceremony of milliAmpere 2's shuttle transit across the main channel in Trondheim (Photo: Kai T. Dragland/NTNU). [7].

and collision avoidance (COLAV) - with companies such as Marine Robotics, DNV GL, Kongsberg, and NTNU - to development of the world's first zero emission, autonomous ship, Yara Birkeland [17, 18]. After a two-year trial period, the Yara Birkeland was put into operation in 2022, intended to operate autonomously [19]. The ship travels 12 nautical miles from the coast between Herøya and Breivik port. The berthing and unberthing procedure will be done without human interaction [20].

In 2022, NTNU tested the world's first urban autonomous passenger ferry, milliAmpere 2. With a capacity of 20 passengers, the project successfully transported passengers over three weeks. Over the course of 6 years, the project has involved over 300 people, including professors, students, and public authorities, establishing the largest research setting dedicated to this field globally [7].

Denmark is another Nordic country that has achieved significant progress in ASVs. In December 2022, a team from the Technical University of Denmark exhibited a driverless catamaran ferry crossing the Limfjord [21]. They developed and trained an Artificial Intelligence (AI) algorithm by gathering data, such as photos of objects and other ships. Thus, utilizing radar, cameras, and sensors to register and avoid objects in proximity to the ferry [21]. However, the system was not run in real-time and they utilized Wärtsilä autodocking system in the demo.

Globally, advancements in ASVs continue to emerge. For instance, software from Orca AI was deployed on a cargo ship to complete a nearly 500-mile voyage, autonomously navigating 99% of the trip [23]. IBM has developed an AI-driven boat, the Mayflower, set to cross the Atlantic Ocean, although it experienced unexpected issues and had to stop in Canada [24]. These developments indicate that the shipping industry is making strides towards ASVs. However, it is challenging to determine the level of autonomy of these ships, as most details remain unpublished.

For a ship to achieve full autonomy, it must successfully navigate through three sailing phases [25]:

- Undocking: Departing from a harbor's quay to open waters.
- Transit: Navigating through a broader water area to reach a designated harbor.
- Docking: Approaching a harbor's docking point from open waters.

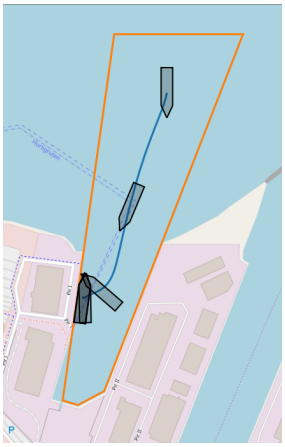
Docking is considered as the hardest phase of sailing, and will therefore be the main focus of this paper. It is a high-risk operation and increasingly challenging when obstacles hinder the path toward the docking point. The docking point is usually in a confined harbor area containing both dynamic and static obstacles. Successful docking requires precise maneuvering at low speeds and accurate allocation of control inputs. This action bears a close dynamic resemblance to dynamic positioning (DP) and can be leveraged to execute a successful docking maneuver.

To safely maneuver within a harbor and account for other vessels, the ASV must demonstrate predictable maneuvers and situational awareness. The Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) provides a comprehensive framework for maritime navigation, outlining the rules and responsibilities to be followed by vessels at sea. These regulations establish guidelines for maintaining a safe distance, determining the right-of-way, and signaling intentions to prevent collisions between vessels. By incorporating the principles and guidelines set forth by COLREGs, ASVs can navigate in compliance with established standards, promoting the safety of both autonomous and manned vessels within harbor areas. Understanding and implementing the COLREGs is therefore essential for the development and deployment of autonomous docking systems, ensuring the seamless integration of ASVs into the existing maritime framework.

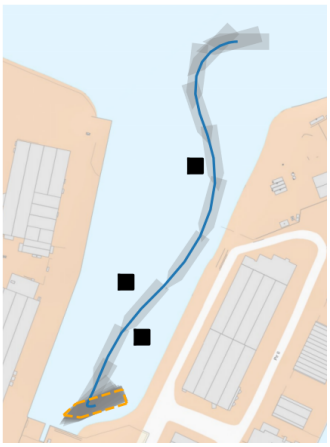
1.1.1 Previous work

Several research papers focus on automating the docking phase. Due to the highly nonlinear nature of the ASV's equations of motion, accurately predicting the model's behavior and addressing the control allocation problem poses significant challenges. Early works such as Rae and Smith (1992) used fuzzy logic control to dock autonomous underwater vehicles, where behavior is modified based on a set of rules. Yamato (1990) pioneered the use of Artificial Neural Networks (ANNs) as an alternative solution to the docking problem.

Subsequent works by Rørvik (2020) expanded the use of ANN by incorporating Deep Reinforcement Learning (DRL) techniques such as Proximal Policy Optimization (PPO) and Deep Deterministic Policy Gradient (DDPG) for docking. In a more recent study, Martinsen et al. (2022) proposed a reinforcement learning-based (RL) model predictive control (MPC) method for trajectory tracking of ASVs, which proved successful in optimal tracking control and control allocation. The proposed method demonstrates effective implementation of optimal tracking control and control allocation. Additionally, the utilization of RL-MPC enhances the robustness of the vessels against model discrepancies



(a) Vessel docking from Martinsen et al. (2019) at Hurtigruten terminal in Trondheim Norway.



(b) Vessel docking with obstacles from Ødven (2022) at Hurtigruten terminal in Trondheim Norway.

Figure 1.2: Results from the methods proposed in Martinsen et al. (2019) and Ødven (2022). Both simulations were performed at Hurtigruten terminal in Trondheim Norway.

and external disturbances.

On the other hand, a recent research effort by Wakita et al. (2022) applied an RL method to reduce collision probability for tracking control during docking, but it exhibited certain weaknesses. The controller was trained solely using obstacles with line segments larger than the ASV's length, which could lead to collisions in real-life scenarios, as many obstacles are smaller. Moreover, the successful docking scenario was vaguely defined, resulting in collisions even at slow speeds. Furthermore, the method was only simulated in a single harbor, limiting the assessment of its robustness and versatility.

The most effective methods utilize optimal control with trajectories planned using convex optimization. However, most of these methods focus solely on docking within an unobstructed and convex harbor environment, devoid of external disturbances. Martinsen et al. (2019) solved the docking procedure as a NOCP, applying spatial constraints to ensure COLAV (Figure 1.2a). This innovative approach laid the groundwork for subsequent full-scale experiments conducted on the autonomous urban ferry, milliAmpere [25, 31]. In these experiments, the ASV utilized harbor maps and sensor data to successfully avoid static obstacles and complete the docking procedure.

The primary disadvantage of representing the docking procedure as an optimization problem is that nonlinear optimization has the propensity to converge to a local optimum. Warm-starting the optimization problem is a possible approach to address this issue. In the research presented by Miyauchi et al. (2022), optimal control is successfully deployed to dock an ASV within a complex harbor geometry. Although promising, this methodol-

ogy's computational complexity proved to be a major drawback, requiring several days to compute a successful simulation. To address this issue, Rachman et al. (2022) utilized a warm-start approach to enhance results. They also introduced a COLAV method to handle convex and non-convex obstacles. Nonetheless, their proposed methodology fell short in effectively dealing with obstacles represented as polygonal holes.

Inspired by the research of Martinsen et al. (2019) and Bitar et al. (2019), Ødven et al. (2022) developed two methods to address the docking problem, particularly in the presence of dynamic obstacles that obstruct the shortest path to the docking point. The first method partitioned the non-convex harbor containing obstacles into feasible convex sets without obstacles, employing the A* search algorithm to calculate the optimal sequence of connecting convex sets for a waypoint-to-waypoint approach. The second method applied triangulation to generate waypoints around the obstacles, similarly utilizing the A* search algorithm for sequence calculation to warm-start the NOCP. This resulted in successful docking without collisions as shown in Figure 1.2b. Odven's methods demonstrated promising results for practical implementation of docking with obstacle avoidance, forming the basis for this Master's thesis to replicate parts of each method for safe docking amid dynamic and static obstacles.

While compliance with COLREGs is crucial to safe navigation, the integration of these rules in ASV control and navigation remains a challenge. A notable research by Ni et al. (2023) proposed a distributed coordinated path planning algorithm for multi-ship encounters in accordance with COLREG rules. However, this method could be computationally intensive and lacks clear guidelines for real-time implementation optimization. The dynamic Bayesian network used to model ship motion uncertainty requires the computation of probabilities for numerous potential ship trajectories, which might be time-consuming.

Eriksen (2019) and Thyri (2022) contributed significantly to this area through their PhD research at NTNU. Eriksen (2019) approached the problem by partitioning it into three segments, utilizing a hybrid architecture to maximize the strengths at different navigation phases. Thyri (2022) developed a COLREGs classification algorithm that determines the encounter type and thus the maneuvering obligations of the ASV in vessel-to-vessel encounters. Both theses offered valuable insights into safe and efficient ASV maneuvering. However, they did not directly address the docking problem, which involves accurately aligning the ASV with the docking station and DP at the endpoint. This thesis will therefore investigate the possibility to incorporate COLREG awareness into an established method for successfully docking ASVs. The level of awareness is determined by the COLREGs classification algorithm, inspired by Thyri (2022).

1.2 Problem description

- **Problem:** Dock a ship without causing a collision with a quay or potential obstacles, whilst adhering to COLREG rules.
- **Proposed solution** Split the problem into several smaller components.

1. Triangulate the archipelago area that includes the initial and desired positions, with the aim of creating waypoints that will safely circumvent any obstacles and land.
2. Find the shortest collision-free course from starting point to the docking point by utilizing the waypoints.
3. Generate a feasible smooth path for the vessel to follow.
4. Generate feasible convex sets along the optimal path that the vessel can maneuver within.
5. Determine the optimal control sequence required to reach the docking point by solving an Optimal Control Problem (OCP) within a Model Predictive Control (MPC) framework.
6. Prevent collisions with other vessels by calculating the Closest Point of Approach (CPA), classifying the encounter according to COLREG rules, and then following the required rules for that specific encounter.
7. Successfully dock the vessel by performing DP at the desired endpoint.

1.3 Contributions

This Master's thesis is a recreation and continuation of the proposed methods from Ødven (2022), Martinsen et al. (2019) and Nordhus (2022). Moreover, elements regarding COLREG awareness from Thyri (2022) have also been recreated. The motive for reconstructing the methods was to improve learning outcomes and provide different views of the docking problem. The following elements from the mentioned articles were recreated:

- Formulating the docking problem as an OCP. Thus, transforming the OCP into a nonlinear programming (NLP) problem using direct collocation.
- Utilizing the methods from Bitar et al. (2019) and Martinsen et al. (2019) to efficiently modify and add spatial constraints to the NLP, representing both the harbor and other obstacles.
- Implementing a constrained Delaunay triangulation and Voronoi diagram to produce safe waypoints. This in turn enables the computation of the shortest collision-free route by exploring these waypoints.
- Automatic generation of convex sets which provide a safe maneuverable area for the ASV, drawing upon the work from Martinsen (2021).
- A COLREGs classifying algorithm utilized to determine which rules are relevant for a specific scenario, inspired by Thyri (2022).

In addition, this thesis introduces several methods for effectively implementing an OCP with COLREGs-aware COLAV, resulting in the following contributions:

- Continuing the progress from Nordhus (2022), further advancements have been made. This includes improving the awareness of the search algorithm to incorporate both COLREGs and areas anticipated to be unoccupied in the future. The resulting path is then utilized to warm-start the OCP.
- A path-smoothing algorithm that guarantees a collision-free route, also taking into account COLREGs, with an easy and flexible implementation strategy.
- An adaptive approach to adjusting the number of time steps in the MPC, aiming to reduce computational time while preserving accuracy.
- An alternative approach to generating convex sets automatically that is computationally feasible and easy to implement.
- Investigation of solving an entire dock-to-dock scenario as an OCP.

1.4 Outline of the report

The thesis consists of 5 chapters:

- Chapter 1 gives an introduction to the docking problem, motivation for making it autonomous, previous work, problem description as well as contributions.
- Chapter 2 presents a thorough explanation of the theory involved in order to solve the docking problem. Hereunder, are vessel kinematics and dynamics, obstacle avoidance, motion planning, and the optimization problem.
- Chapter 3 presents the methodology and utilized library.
- Chapter 4 presents the simulations and results achieved and discusses the subsequent implications of solving the docking problem.
- Chapter 5 concludes the thesis.
- Appendix A and B outline the vessel model and the different parameter values applied in the simulations, respectively.

2

Background theory and methods

This chapter presents the relevant theory used in this thesis to solve the docking problem. Firstly, a definition of the vessel model and thrust allocation. Secondly, convex polygons and a method for traversing around objects within a convex set are presented. This is essential for defining which area the vessel can maneuver within. Thereby, a presentation of how to plan a feasible path while adhering to the conventions at sea. Lastly, an introduction to the optimization problem and a method used to solve such a problem numerically. It should be noted that this thesis is a continuation of the project report, Nordhus (2022), and it will therefore draw on the same theoretical background.

2.1 Vessel model

2.1.1 Kinematics and dynamics

A three-dimensional rigid body has 6 degrees of freedom (DOF). The same goes for maneuvering a marine vessel. It can move freely along three perpendicular axes, forward/backward (surge), left/right (sway), and up/down (heave), also known as $\{x, y, z\}$ in classic mathematics. Additionally, it can freely rotate around each of these axes'. Rotation around x, y, and z are normally represented as yaw, pitch, and roll, respectively. This is illustrated in Figure 2.1. The notation for the kinematics is represented in table 2.1 and is similar to the notation used by the Society of Naval Architects and Marine Engineers (SNAME).

When a marine vessel is docking, heave motions as well as pitch and yaw rotations are normally small. Consequently, the dynamics of the vessel can be simplified to 3-DOF as a sufficiently good approximation.

In order to represent a vessel's coordinates by these notations, it is essential to use geographical reference frames. Essentially, two frames are needed. One frame for representing the marine craft and one for representing the marine craft's movement relative to the

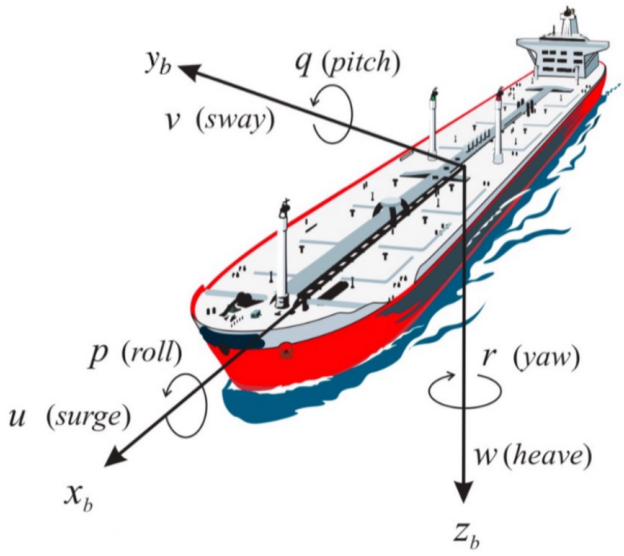


Figure 2.1: Marine craft maneuverability with 6 DOF. Illustration from Fossen (2011).

	BODY			NED
DOF		Forces and moments	Linear and angular velocities	Positions and Euler angles
1	Motions in the x_b-direction (surge)	X	u	x^n
2	Motions in the y_b-direction (sway)	Y	v	y^n
3	Motions in the z_b-direction (heave)	Z	w	z^n
4	Rotation about the x_b-direction (roll)	K	p	ϕ
5	Rotation about the y_b-direction (pitch)	M	q	θ
6	Rotation about the z_b-direction (yaw)	N	r	ψ

Table 2.1: SNAME notation for kinematics of a marine vessel [6].

earth. An earth-centered inertial (ECI) frame, where Newton's laws apply, can be utilized to depict the earth. However, it is not necessary to utilize a geographical frame of the entire earth when maneuvering a marine craft over a small area. Therefore, the ECI is simplified to the North-East-Down frame. In Fossen (2011), the reference frames are defined as:

- **The North-East-Down (NED) reference frame:** $\{n\} = \{x_n, y_n, z_n\}$, is defined relative to the earth reference ellipsoid, usually as the tangent plane to the ellipsoid. The x_n axis points to true North. The y_n axis points East. The z_n axis points downward normal to the Earth's surface.
- **The Body-Fixed (BODY) reference frame:** $\{b\} = \{x_b, y_b, z_b\}$ is a moving coordinate system fixed to the marine craft body in order to represent the vessel inertia and dynamics. The x_b axis points from aft to fore. The y_b axis points starboard. The

z_b axis points from top to bottom.

Rotation matrices are used in order to relate one reference frame to another (while keeping the vessels kinetics) without changing the geometry of the vessel. The transformation between different coordinate systems results in different rotation matrices. The rotation matrix

$$\boldsymbol{J}_\psi = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.1}$$

where ψ is the heading angle, is used to transform the kinematics and kinetics from the BODY frame to the NED frame.

The dynamics of a marine craft consist of kinematics and kinetics. Kinetics studies the motions and forces involved, whereas kinematics studies motion without considering what forces caused them. A 3-DOF marine craft's dynamics can be described by the equations of motion in vectorial form as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_\psi \boldsymbol{\nu}, \tag{2.2a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}\boldsymbol{\nu} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{\tau}, \tag{2.2b}$$

where $\boldsymbol{M} \in \mathbb{R}^{3\times3}, \boldsymbol{C} \in \mathbb{R}^{3\times3}$, $\boldsymbol{D} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{\tau} \in \mathbb{R}^3$ are the inertia matrix, Coriolis matrix, dampening matrix, and control input vector, respectively.

2.1.2 Thrust allocation

In Fossen (2011), a marine craft's control inputs, $\boldsymbol{\tau}$, are determined by the thrust configuration matrix $\boldsymbol{T}(\boldsymbol{\alpha})$. This matrix allocates the distinct thrust forces $\boldsymbol{f}$ applied to each of the motors given an angle $\boldsymbol{\alpha}$. Forces acting in the surge and sway (x, y) direction cause momentum about the yaw (z axis). Therefore, are the control inputs on a 3-DOF marine vessel represented by

$$\boldsymbol{\tau} = \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{f} = \begin{bmatrix} X \\ Y \\ N \end{bmatrix}, \tag{2.3}$$

where X and Y are forces in surge and sway direction and N is the momentum generated about the yaw axis, as presented in table 2.1. The forces and moments are distinct for each thruster and can be further specified for a single thruster, i, as

$$\boldsymbol{T}_i(\boldsymbol{\alpha})\boldsymbol{f}_i = \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{y,i}l_{x,i} - F_{x,i}l_{y,i} \end{bmatrix} = \begin{bmatrix} f_i\cos(\alpha_i) \\ f_i\sin(\alpha_i) \\ f_i(l_{x,i}\sin(\alpha_i) - l_{y,i}\cos(\alpha_i)) \end{bmatrix}. \tag{2.4}$$

Selecting the correct angles $\boldsymbol{\alpha}$ and force $\boldsymbol{f}$ formulates the thrust allocation problem. A vessel is fully-actuated if the number of control inputs is equal to or greater than the degrees of motion. Moreover, the allocation problem depends on the thruster characteristics. This thesis will only consider one type of thruster:

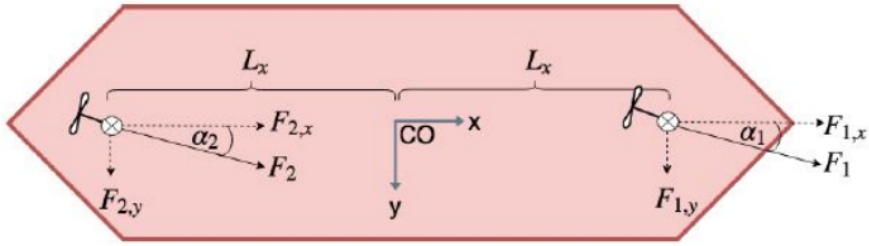


Figure 2.2: Thruster configurations of the milliAmpere utilizing two azimuth thrusters. Illustration from Pedersen (2019).

- **Azimuth thrusters:** are propellers placed in a pod that can rotate typically at any angle, α, about the normal (z) axis. Therefore, they will produce a force in both the lateral and longitudinal (x, y) direction.

The research vessel milliAmpere, depicted in Figure 2.2, is a fully-actuated monohull vessel located at SFI Autoship. Although the ferry is designed to carry a maximum of six people, it is not authorized for commercial passenger transportation. Its primary purpose is to function as a testing ground for system components such as motion control, autonomy, and various sensor configurations (Brekke et al. (2022)). The vessel is specified further in Appendix A.

2.2 Computational geometry

A search algorithm must have a feasible search space in order to find a optimal solution. Computational geometry and a grid-based approach are two common methods for partitioning the continuous search space into a discrete search space. Grid-based approaches are simple and easy to implement by dividing the search space into a grid of cells. However, the grid resolution and connectivity have a significant impact on the efficiency of this approach. If the grid resolution is too low, large cells, the optimal path may not be able to avoid obstacles. On the other hand, if the resolution is too high, the search space can become too large by increasing the computational time required to find the optimal path. Connectivity issues occur typically when the resolution is too low, where some cells may not be connected to each other since obstacles occupy part of them.

Utilizing computational geometry, on the other hand, to partition the continuous search space can address some of these issues. This approach allows for more flexibility, as the cell shapes may take obstacles into account. Additionally, providing more precise information about the environment. This approach can also reduce the computational requirements by reducing the number of necessary cells.

2.2.1 Convex set

A set in Euclidean space is convex if any given point inside the set can be connected by a line segment that will always stay inside the set. Similar to Martinsen et al. (2019), this thesis makes use of properties given by a convex set and can be referred to as a convex polygon due to its shape. Convex polygons can easily be implemented in an optimization problem since they can be described as a solution to the linear inequality constraints

$$\boldsymbol{A}v_i \leq \boldsymbol{b},\ \forall i \in [1, \ldots, m] \tag{2.5}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times 3}, \boldsymbol{b} \in \mathbb{R}^{m\times 1}, \boldsymbol{v}_i \in \mathbb{R}^2$ and m is the number of vertices in the polygon.

2.2.2 Automatic convex set generation

Convex sets are essential to the optimization problem since they decide where it is safe for the vessel to maneuver. Islands and harbor areas are typically non-convex. Therefore, it is important to generate convex sets that the vessel safely can travel within. There exist many different methods for generating convex hulls within complex surroundings such as Bergman et al. (2020) and Martinsen (2021). The method proposed in Martinsen (2021), inspired by [40, 41], involves 5 steps:

1. Expanding an ellipse from a center point $\boldsymbol{p}_c$ until it intersects with an environmental constraint.
2. Find the point, $\boldsymbol{p}_{ab}$, along the intersecting constraint that is closest to $\boldsymbol{p}_c$.
3. Obtain the tangent line to the expansion ellipse that intersects with the point $\boldsymbol{p}_{ab}$.
4. Repeat the ellipse expansion and linear constraint generation process for each line segment making up the environmental constraints.
5. Remove redundant constraints by obtaining the final convex inner approximation of the non-convex area.

The execution of these steps involves solving various optimization problems. Firstly, an optimization problem is solved to determine the closest point $\boldsymbol{p}_{ab}$. Next, a linear equality is solved to find the tangent line. Lastly, a Linear Programming problem is solved to eliminate redundant constraints.

To optimize the constraints, a normalization technique is applied by multiplying each constraint row with a normalizing factor. This ensures that the resulting matrix has unit length rows, facilitating meaningful physical interpretations and enabling the addition of margins or relaxation of the constraints.

Overall, the presented method provides a comprehensive approach for computing the convex inner approximation set for the automatic docking and berthing of ASVs. The vessel-centered approach, utilization of line segments as obstacles, and optimization techniques contribute to the efficient and effective computation of the convex inner approximation, ensuring safe navigation in the presence of environmental obstacles.

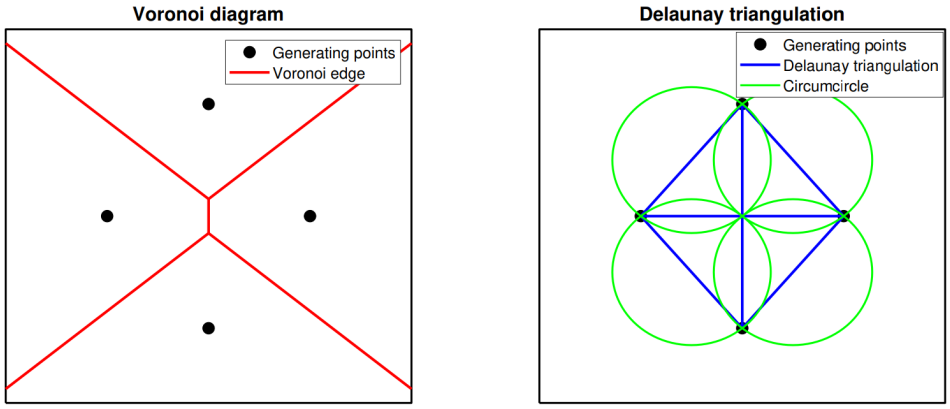


Figure 2.3: Illustration of the duality between Voronoi diagram and Delaunay triangulation [4].

2.2.3 Triangulation

There are several existing techniques for partitioning non-convex 2D space into smaller convex polygons. This thesis will focus on two popular methods for partitioning non-convex polygons; Delaunay triangulation and Voronoi diagram. They represent the same concept from different points of view.

Voronoi diagram

Given a set of points $\boldsymbol{q}_i \in \mathbb{S}$ for $i = [1, \ldots, n]$, and n is the number of points, the Voronoi diagram is a method for finding which region or cell is closest to each point $\boldsymbol{q}_i$. The edge of each cell provides information about where there is an equal distance between the points in $\boldsymbol{q}$. A Voronoi cell $\boldsymbol{\nu}_{q_j}$ can then be defined as

$$\boldsymbol{\nu}_{q_j} = \{x \in \mathbb{R}^2 \mid ||x - \boldsymbol{q}_j|| \leq ||x - \boldsymbol{q}_i||, \ \forall_{\boldsymbol{q}_i, \boldsymbol{q}_j} \in \mathbb{S}, \ i \neq j\}, \tag{2.6}$$

where x is an arbitrary point and $\boldsymbol{q}_j$ is the point generating a Voronoi cell.

Simplified, the edges of the Voronoi cells are generated by creating a line segment between the neighboring points in $\boldsymbol{q}$. Then, the perpendicular bisector of that line will represent the edge of the Voronoi cell. Any point along this line will be at an equal distance to each point $\boldsymbol{q}_i$.

Delaunay triangulation

Delaunay triangulation is the dual problem of the Voronoi diagram. Let's define the same set of points as in the Voronoi diagram, $\boldsymbol{q}_i \in \mathbb{S}$ for $i = [1, \ldots, n]$, and n is the number of points. The Delaunay triangulation is generated by creating a line segment between each neighboring point, whereas the circumcircle around each generated triangle cannot contain any of the other generating points. Figure 2.3 illustrates each of the triangulation

methods. It is visible that the Voronoi edges are the perpendicular bisector of the Delaunay triangulation, highlighting the duality of the two methods.

2.3 Motion planning

2.3.1 Convention on the International Regulations for Preventing Collisions at Sea (COLREGs)

In order to plan a safe, collision-free trajectory towards the desired goal, while avoiding dynamic obstacles, it is highly beneficial to have a sense of the obstacles' movement. The Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) is an international agreement that establishes a set of rules and regulations designed to prevent collisions between vessels at sea. The convention was first adopted in 1972 and has since been updated to reflect advances in technology and changes in the maritime industry.

The main goal of this convention is to ensure the safety of all vessels and crew on the water and to prevent collisions that can cause damage, injuries, or loss of life. By providing a set of clear and consistent rules that are recognized and followed by all seafarers, the convention helps to minimize the risk of accidents and promote safe and efficient maritime operations. However, the rules only consider instances with two vessels in proximity to each other.

The COLREGs include 41 rules divided into six sections. However, only specific rules may be of relevance when maneuvering an autonomous ship within a confined area. Herein section B, steering and sailing, including the conduct of vessels in any condition of visibility, and the conduct of vessels in sight of one another. Rules 5-9 are applicable since they cover requirements on visibility, safe speed, risk of collision, action to be taken to avoid a collision, and preferred maneuver within a narrow channel, respectively. Moreover, rules 13-17 are also applicable as they deal with overtaking, head-on situations, crossing situations, and the action of the give-way and stand-on vessel, respectively. On the other hand, rule 10 is less relevant, as it is specific to traffic lanes at sea, which this thesis will not focus on. Rule 12 is also less relevant, as it is specific to sailing vessels. Rule 18 is less applicable, as the autonomous ship may not have an obligation to keep out of the way of other vessels, but it still has a responsibility to avoid a collision. The relevant rules for an autonomous ship within a harbor area are described in Table 2.2 [11, 42, 43].

Rule	Description
5	**Look-out:** Every vessel shall at all times maintain a proper lookout by sight and hearing as well as by all available means appropriate in the prevailing circumstances and conditions so as to make a full appraisal of the situation and of the risk of collision.
6	**Safe speed:** Every vessel shall at all times proceed at a safe speed so that she can take proper and effective action to avoid collision and be stopped within a distance appropriate to the prevailing circumstances and conditions.
7	**Risk of collision:** Every vessel shall use all available means appropriate to the prevailing circumstances and conditions to determine if risk of collision exists. If there is any doubt such risk shall be deemed to exist.
8	**Action to avoid collision:** If a vessel needs to take action to avoid collision, it should be large enough to be readily observable of other ships, implying that series of small alternations in speed and/or course should not be applied. Course changes should be prioritized over speed changes if there are enough free space available, and that maneuvers must be made in ample time.
9	**Narrow channels:** A vessel proceeding along the course of a narrow channel or fairway shall keep as near to the outer limit of the channel or fairway which lies on her starboard side as is safe and practicable.
13	**Overtaking:** Any vessel overtaking another vessel shall keep out of the way of the vessel being overtaken. A vessel approaching another vessel from a direction of more than 22.5 *deg* abaft her beam is an overtaking vessel. Any subsequent alternation of bearing between the two vessels shall not relieve the overtaking vessel of the duty of keeping clear of the overtaken vessel until she is finally past and clear.
14	**Head-on situation:** When two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other vessel.
15	**Crossing situation:** When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel.
16	**Action by give-way vessel:** Every vessel which is directed to keep out of the way of another vessel shall, so far as possible, take early and substantial action to keep well clear.
17	**Action by stand-on vessel:** Where one of two vessels is to keep out of the way, the other shall keep her course and speed. The latter vessel may take action to prevent collision if it is apparent that the vessel required to keep out of the way is not taking appropriate action.

Table 2.2: COLREG rules 5-9, 13-17 and their specifications.

2.3.2 Collision risk assessment

Evaluating the collision risk is an important aspect of path planning when dealing with dynamic obstacles. Calculating the closest point of approach (CPA) is a common method used to assess the collision risk of the current path of the vessel. The CPA is found by calculating the time to CPA, t_{CPA}, and distance at CPA, d_{CPA}, given by

$$t_{\text{CPA}} = \frac{(\boldsymbol{p}_{\text{OS}} - \boldsymbol{p}_{\text{TS}}) \cdot (\boldsymbol{v}_{\text{OS}} - \boldsymbol{v}_{\text{TS}})}{\|\boldsymbol{v}_{\text{OS}} - \boldsymbol{v}_{\text{TS}}\|}, \tag{2.7}$$

$$d_{\text{CPA}} = \|\boldsymbol{p}_{\text{OS}} + \boldsymbol{v}_{\text{OS}} t_{\text{CPA}} - (\boldsymbol{p}_{\text{TS}} + \boldsymbol{v}_{\text{TS}} t_{\text{CPA}})\|, \tag{2.8}$$

where $\boldsymbol{p}$ and $\boldsymbol{v}$ are the position and velocity vectors of the ownship (OS) and target ship (TS). The norms are the Euclidean norm, defined by

$$\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \ldots + x_n^2}. \tag{2.9}$$

In Kuwata et al. (2014) and Candeloro et al. (2017), the motion planner evaluates if the COLREGs rules are relevant for a moving vessel by checking whether there is a possibility of a collision or near collision in the immediate future. The evaluation is done by checking if

$$0 \leq t_{\text{CPA}} \leq t_{risk} \quad \cap \quad d_{\text{CPA}} \leq d_{risk}, \tag{2.10}$$

where t_{risk} and d_{risk} are design parameters chosen according to how aggressive or passive a vessel should be when observing other vessels.

2.3.3 Path planning

Path planning is a critical task in robotics that involves finding a feasible path to navigate from a starting point to a goal point while avoiding obstacles and minimizing various criteria such as time, energy, or distance. Solving the path planning problem is generally considered an NP-hard problem [46, 47]. NP-hard problems are problems that are at least as hard as the hardest problems in the class NP (nondeterministic polynomial time), meaning they cannot be solved in polynomial time by any known algorithm. Solving the path planning problem optimally requires exploring an exponentially large search space of possible paths, making it computationally intractable for many robotic applications. On the other hand, several methods exist for finding a feasible path efficiently for practical purposes and can be classified typically into three different approaches.

- **Algorithmic:** There are typically two algorithmic approaches, search-based methods, and sampling-based methods. Search-based algorithms, such as A*, Dijkstra's algorithm, and depth-first search, search for a path by exploring a search space, whereas sampling-based algorithms, such as Rapidly Exploring Random Tree (RRT), probabilistic roadmap (PRM), and potential fields, sample the environment to generate a set of feasible paths.
- **Model-based:** Model-based path planning utilizes a system's dynamics, the operational environment, and the planning objective. This knowledge can be used to simulate several trajectories and evaluate them based on the desired objective.

- **Environmental:** Most path planning approaches require a definition of the environment which they are operating within, such as 2D or 3D, static or dynamic, known or unknown. However, some methods exist, such as Simultaneous Localization And Mapping (SLAM), which allows the robot to simultaneously explore and build a map of its environment as it moves. Other methods, such as RRT*, can adapt to changes in the environment during planning by dynamically updating the tree structure.

Search algorithm

In this thesis, the operational environment is assumed to be known, and computational geometry will be utilized to generate a statically discrete state space where the popular search algorithm, A*, can find the shortest feasible path.

The A* algorithm is an extension of Dijkstra's algorithm that uses heuristics to guide the search from the starting node to the goal node. Dijkstra's algorithm selects the closest nodes, while A* additionally uses a heuristic estimate of the remaining cost to the goal node to guide the search from the starting node to the goal node. The cost function is defined as $f(n) = g(n) + h(n)$, where $g(n)$ is the travel cost from the current node to the neighboring node n, and $h(n)$ is the heuristic estimate of the cost from node n to the goal node. The heuristic function must be admissible, meaning it never overestimates the actual cost to the goal node.

The A* algorithm repeatedly selects the node with the lowest cost function and expands nodes based on their costs and heuristic estimates. As it progresses, it updates the cost values of nodes based on the costs of their neighbors and the heuristic estimates. This process continues until the goal node is reached or there are no more nodes to evaluate. A* guarantees both optimality and completeness, meaning that it will find the shortest path if one exists. The pseudocode of the A* algorithm from Sharma et al. (2012) is shown in **Algorithm 1**.

Computational efficiency is an important aspect of any search algorithm that is not to be used offline. The A* algorithm's time complexity depends on the assumptions about the state space and the efficiency of the heuristic function. The relative error of the heuristic is defined as

$$\epsilon = \frac{h^* - h}{h^*}, \tag{2.11}$$

where h^* is the actual remaining cost. In a practical sense, the relative error is constant or growing, since the error, $h^* - h$, is proportional to the path cost. The computational cost of the A* algorithm is, therefore, $O(b^{\epsilon d})$, where b and d are the branching factor and depth of the solution, respectively. Computation time is not, however, A*'s main drawback. Since it keeps all generated nodes in memory, A* usually runs out of space long before it runs out of time (Russell and Norvig (2010)).

Algorithm 1: A* algorithm

Input:
A graph with source node, *startNode*, and goal node, *endNode*

Initialize
Initialize an empty $open$ and $closed$ list
$g(\textit{startNode}) = 0$
$h(\textit{startNode}) =$ Distance(*startNode, endNode*)
$f(\textit{startNode}) = g(\textit{startNode}) + h(\textit{startNode})$
Append *startNode* to $open$

while $open \neq \varnothing$ **do**
  currentNode $\leftarrow node \in open$ with lowest cost $f(node)$
  if *currentNode* = *endNode* **then**
    return
  $open$.pop(*currentNode*)
  $closed$.append(*currentNode*)
  for *every neighborNode of currentNode* **do**
    if *neighborNode* $\in closed$ **then**
      continue
    cost $\leftarrow$ g(currentNode) + Distance(*currentNode, neighborNode*)
    if *neighborNode* $\in$ *open* ***and*** *cost* $<$ *g(neighborNode)* **then**
      open.pop(*neighborNode*)
    if *neighborNode* $\in$ *closed* ***and*** *cost* $<$ *g(neighborNode)* **then**
      closed.pop(*neighborNode*)
    if *neighborNode* $\notin$ *open* ***and*** *neighborNode* $\notin$ *closed* **then**
      open.append(*neighborNode*)
      g(neighborNode) = *cost*
      h(neighborNode) = Distance(*currentNode, endNode*)
      f(neighborNode) = *g(neighborNode)* + *h(neighborNode)*
return failure

2.3.4 Path smoothing

A path normally contains several waypoints, where a waypoint is defined as the Cartesian coordinates (x, y). In this thesis, a waypoint i is defined as a pose, $\boldsymbol{\eta}_i = [x_i, y_i, \psi_i]^\top$, which in addition to the Cartesian coordinates, includes the heading of the vessel. Additionally, the waypoints contain information about the desired forward speed, U_i the vessel should have when passing through waypoint i.

Finding the shortest path with a search algorithm may render a path that is infeasible or illogical for a vessel to follow due to sharp turns. An often-used workaround method within marine guidance is waypoint-switching with a circle of acceptance. In this approach, a vessel moves along a piecewise linear path, and if the vessel is within a circle

of acceptance with radius R_i around waypoint (x_i, y_i) the next waypoint (x_{i+1}, y_{i+1}) is selected. Another method applies only the along-track distance, x_e (Figure 2.4), to switch waypoints (Breivik and Fossen (2009)). It is advantageous to not have any restrictions on the along-track error, y_e, if the waypoints do not have any inherent value and are only used to define a piecewise linear path. Alternatively, a feasible path can be generated by utilizing smoothing techniques. There exist several methods for generating a smooth path from waypoints or control points, such as polynomial fitting, Bézier curves, splines, or Dubin's curve, some illustrated in Figures 3.9 and 3.10.

Bézier curves

Bézier curves are defined by a set of $n+1$ control points, $P_0, P_1, \ldots, P_n$, where the curve always passes through the first and the last control point. Bernstein polynomial is utilized in order to generate the Bézier curve

$$C(t) = \sum_{i=0}^{n} P_i B_i^n(t), \quad t \in [0,1], \tag{2.12}$$

where t is a normalization value indicating the distance along the path and $B_i^n(t)$ is the Bernstein polynomial given by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}. \tag{2.13}$$

De Casteljau's algorithm is a recursive method for evaluating Bézier curves. The algorithm computes points on the curve by recursively computing the convex combination of pairs of adjacent control points. The algorithm starts by linearly interpolating between each pair of adjacent control points to generate a new set of points and then repeats the process with the new set of points until a single point is obtained. The final point is a point on the Bezier curve.

The benefit of utilizing this method is that the control points can be placed to generate the curve with desired characteristics. Since de Casteljau's algorithm does not assure a collision-free path, more control points can be added to get a curve with desired features. However, the placement of these control points can be difficult, and increasing the number of control points increases the computational cost equivalently.

B-Spline

B-Splines is a generalization of a Bézier curve. Unlike a Bézier curve, splines can easily be controlled by placing more control points along the desired path without increasing the computational cost noticeably. In order to ensure a collision-free path, control points can easily be placed on the original path where it is in proximity to an obstacle. A B-Spline is found with $n+1$ control points, $P_0, P_1, \ldots, P_n$, by computing

$$C(t) = \sum_{i=0}^{n} P_i B_i^n(t), \quad t \in [0,1], \tag{2.14}$$

where t is a normalization value indicating the distance along the path and $B_i^n(t)$ is the basis function given by

$$B_i^n(t) = \frac{t - t_i}{t_{i+n} - t_i} B_i^{n-1}(t) + \frac{t_{i+n+1} - t}{t_{i+n+1} - t_{i+1}} B_{i+1}^{n-1}(t) \tag{2.15}$$

Dubin's curve

A common approach to generating a smooth path in marine guidance is by utilizing straight lines and circle arcs to connect the waypoints, also known as Dubin's curve, which can be summarized as (Fossen, Dubins (2011, 1957)):

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

This method generates a path with discontinuous curvature with a desired curvature that can consider the turning radius of the vessel. However, a drawback with this method compared to splines is the jump in desired yaw rate which produces a small offset during cross-tracking.

2.3.5 Path following

In marine guidance, path following refers to the task of steering a vessel along a predefined path by controlling the heading and speed in order to minimize deviation. There are many methods that can be used for path following, such as Line-of-Sight (LOS), vector-field, pure-pursuit (PP), or constant bearing (CB) to name a few [50].The LOS guidance law, depicted in Figure 2.4, is relatively simple and works well on straight paths or when the vessel is traveling at high speed. In this thesis, the vessel will follow a curved path at a relatively low speed. In order to minimize the deviation from the desired path, this thesis will in, addition to the LOS guidance law, also consider a time-varying lookahead distance guidance law and a PP guidance law from Lekkas and Fossen (2012) and Fossen (2011). Convergence to the desired path is achieved by choosing the desired heading as

$$\psi_d = \psi_{\text{LOS}} - \arctan\left(\frac{y_e}{\Delta}\right), \tag{2.16}$$

where y_e and $\Delta > 0$ are the *cross-track error* and *lookahead distance*, respectively. The angle $\psi_{\text{LOS}} \in (-\pi, \pi)$ is the *path-tangential angle* expressed as α_k in Figure 2.4. The angle is given by

$$\psi_{\text{LOS}} = \arctan\left(\frac{y_{\text{LOS}} - y}{x_{\text{LOS}} - x}\right), \tag{2.17}$$

where $p = [x, y]^\top$ is the vessel's current Cartesian coordinates and $p_{\text{LOS}} = [x_{\text{LOS}}, y_{\text{LOS}}]^\top$ is the point on the desired path that the vessel should be pointing at.

In order to optimize the convergence towards the path, a time-varying lookahead distance can be chosen as

$$\Delta(y_e) = (\Delta_{max} - \Delta_{min})e^{-\gamma|y_e|} + \Delta_{min}, \tag{2.18}$$

where $\gamma > 0$ is a design parameter to ensure convergence, and Δ_{max} and Δ_{min} are the maximum and minimum chosen values for Δ, respectively. A time-varying lookahead distance returns a small Δ, and therefore a more vigorous behavior, if the cross-track error is considerable, and a large Δ when the vessel is close to the path. This maneuver minimizes the chances of overshooting when converging to the desired path [10].

In addition to a waypoint containing information about the desired pose, it is also common to include information about the desired forward speed the vessel should have when passing the vessel. The PP guidance law is one way of deciding the vessel's speed through waypoint i. The desired velocity $\boldsymbol{v}_d = [u_d, v_d]^\top$ at waypoint i is found by

$$\boldsymbol{v}_{d,i} = -\kappa \frac{\tilde{\boldsymbol{p}}}{\|\tilde{\boldsymbol{p}}\|}, \tag{2.19}$$

where $\kappa > 0$ and $\tilde{\boldsymbol{p}} := \boldsymbol{p}_{\text{i}} - \boldsymbol{p}_n$ is the distance between waypoint i and waypoint n in a path with n waypoints.

CB guidance utilizes similar geometry as PP guidance whilst taking into account the velocity vector of a target. The goal of this approach is to reduce the LOS rotation rate, closing in on the target in a direct collision course. The desired CB velocity is then given by

$$\boldsymbol{v}_d = \boldsymbol{v}_t + \boldsymbol{v}_a \tag{2.20}$$

$$\boldsymbol{v}_a = -\kappa \frac{\tilde{\boldsymbol{p}}}{\|\tilde{\boldsymbol{p}}\|} \tag{2.21}$$

where $\boldsymbol{v}_a = [u_a, v_a]^\top$ is the approach velocity vector which gives the desired approach speed $U_a = \|\boldsymbol{v}_a\|$. From Breivik and Fossen (2007), κ is given as

$$\kappa = U_{max} \frac{\|\tilde{\boldsymbol{p}}\|}{\sqrt{(\tilde{\boldsymbol{p}})^\top \tilde{\boldsymbol{p}} + \Delta^2}}, \tag{2.22}$$

where $U_{max} > 0$ is the maximum approach speed toward the target, waypoint n. In this thesis, the target point is a stationary waypoint and the CB guidance becomes therefore equal to the PP guidance.

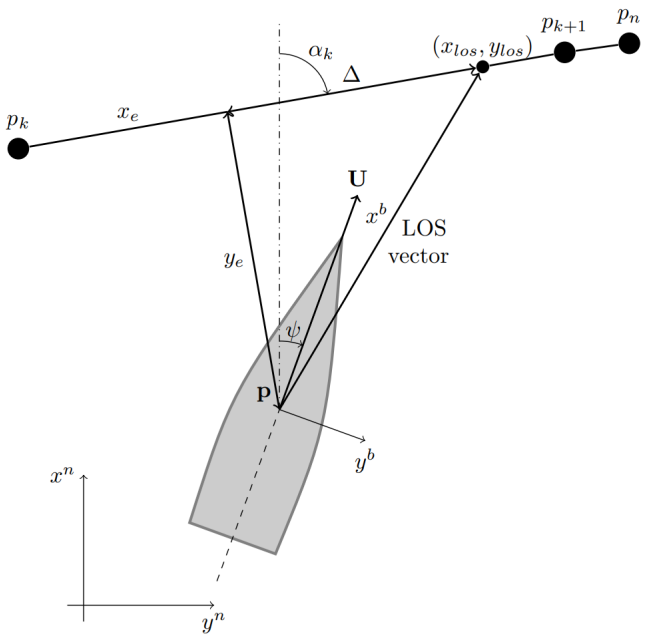


Figure 2.4: Illustration of the geometry and variables utilized in LOS guidance. Courtesy of Lekkas and Fossen (2012).

2.4 Optimization problem

An optimization problem is the problem of finding the best solution from all feasible solutions, typically for dynamic systems. A dynamic system describes a current state and how it will develop over time. In order to discover the optimal ways to control a dynamic system, the mathematical field of optimal control theory was developed. Optimal control theory solves optimization problems by determining a control law that can be applied to the dynamic system. It can be applied to a range of disciplines such as business, economics, and engineering. For this thesis, optimal control is used to find the optimal trajectory and control inputs for a DP controller to dock successfully.

An optimization problem typically consists of four components:

- **An objective function** that is desired to minimize or maximize.
- **Decision variables**, commonly control and state variables related to each other by a set of differential equations.
- **Constraints** that limits and describes the physical system.
- **A mathematical model** that represents the problem to be solved, typically in the form of equations and inequalities.

The optimization problem is called a *nonlinear programming* (NLP) *problem* if the objective function is nonlinear and/or the feasible region is determined by nonlinear constraints [53]. A general NLP can be expressed as

$$\min_{w} \quad \mathrm{J}(\boldsymbol{w}) \tag{2.23a}$$

$$\text{subject to} \quad \boldsymbol{g}_1(\boldsymbol{w}) \leq 0 \tag{2.23b}$$

$$\boldsymbol{g}_2(\boldsymbol{w}) = 0, \tag{2.23c}$$

where J is the objective function, $\boldsymbol{w}$ is the decision variables, and $\boldsymbol{g}_1(\boldsymbol{w})$ and $\boldsymbol{g}_2(\boldsymbol{w})$ are the inequality and equality constraints, respectively. In certain scenarios, the constraints may be too strict to find a good solution to the optimization problem. Slack variables may then be added to allow for certain constraints to be violated. The slack variables $s \geq 0$ are added to the inequality constraints $\boldsymbol{g}_1(\boldsymbol{w}) \leq 0$, replacing it with an equality constraint $\boldsymbol{g}_1(\boldsymbol{w}) + s = 0$, and a non-negativity constraint [54].

Solving the docking problem can be described as a nonlinear optimal control problem (NOCP) due to the highly nonlinear nature of the ASV control problem. The NOCP can be formulated as in Kirches et al. (2012) as

$$\min_{x(\cdot),u(\cdot)} \quad \int_{t_0}^{t_0+T} L(\boldsymbol{x}(t), \boldsymbol{u}(t))dt \tag{2.24a}$$

$$\text{subject to} \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{2.24b}$$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad t \in [t_0, t_0 + T], \tag{2.24c}$$

$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq 0, \quad t \in [t_0, t_0 + T], \tag{2.24d}$$

where $\boldsymbol{x}(\cdot)$ and $\boldsymbol{u}(\cdot)$ are the state and control variables, L is the cost function, t_0 is the initial time, and T is the time horizon. In order to solve the NOCP, the initial constraints (2.24b), state dynamics (2.24c), and path constraints (2.24d) have to be satisfied.

There are commonly used three different methods for solving an optimal control problem (OCP) numerically:

- **Dynamic Programming** breaks down the optimization problem into smaller subproblems and stores the solution to each sub-problem so that it is only solved once. Methods for solving this problem numerically suffer from Bellman's "curse of dimensionality" and are restricted to a small state space [56, 57].
- **Indirect methods** apply Pontryagin's maximum principle [58] which provides an efficient way of evaluating the gradient cost with respect to the control input, thereby indirectly optimizing the controls numerically. Even though this approach is often very accurate, challenges occur from implementing the state constraints and the need to formulate first-order necessary conditions for every problem instance [59].
- **Direct methods** solve the optimal control problem for purely continuous, nonlinear systems based on a discretization with respect to time [60]. They are less accurate than indirect methods, but benefit from flexible handling of constraints and being numerically stable. This bodes well for a practical implementation of the OCP.

Direct methods can be split into two main classes; sequential methods and simultaneous methods. Sequential methods, such as direct single shooting [61], are almost certain to fail when solving complicated problems such as an NLP problem. It only uses control inputs as optimization variables to calculate the objective and constraints. Resulting in small problems with no structure. Simultaneous methods, such as direct multiple shooting and direct collocation [62, 63], use both control inputs and states as optimization variables to directly represent the state trajectory. These methods only satisfy the dynamics constraint at specific points along the trajectory [64]. The system model is included as equality constraints, resulting in large optimization problems with very structured constraints. Direct multiple shooting and direct collocation offer the same optimization stability. Even though direct multiple shooting has a more flexible approach than direct collocation, it solves the problem slower. This is due to the direct collocation method solving the numerical integrations as a part of the optimization problem. This thesis will therefore utilize the direct collocation for solving the NLP, similar to Martinsen et al. (2019) and Ødven et al. (2022).

2.4.1 Direct collocation method

The direct collocation method discretizes the controls and states on a grid t_k with $k \in [0, N]$ for the OCP. A set collocation points are chosen for each collocation interval $[t_k, t_{k+1}]$ for $t_{k,i}$, where $i \in [0, d]$. A polynomial $\boldsymbol{p}_k(t, v_k) \in \mathbb{R}^n$ with coefficients $\boldsymbol{v}_k \in \mathbb{R}^{n_x(d+1)}$ is used to approximate the trajectory of each state on the collocation interval. Continuity across the interval boundaries is required, i.e. that

$$\boldsymbol{p}_k(t_{k+1}, \boldsymbol{v}_k) - \boldsymbol{s}_{k+1} = \boldsymbol{0}, \tag{2.25}$$

must hold for $k \in [0, N]$, where $\boldsymbol{s}_k$ is the discrete states on the grid points t_k. In order to integrate the system dynamics over a collocation interval, the following equations must be solved [65]:

$$\boldsymbol{c}_k(\boldsymbol{v}_k, \boldsymbol{s}_k, \boldsymbol{q}_k) = \begin{bmatrix} \boldsymbol{v}_{k,0} - \boldsymbol{s}_k \\ \dot{\boldsymbol{p}}_k(t_{k,1}, \boldsymbol{v}_k) - \boldsymbol{f}(\boldsymbol{v}_{k,1}, t_{k,1}, \boldsymbol{q}_k) \\ \vdots \\ \dot{\boldsymbol{p}}_k(t_{k,d}, \boldsymbol{v}_k) - \boldsymbol{f}(\boldsymbol{v}_{k,d}, t_{k,d}, \boldsymbol{q}_k) \end{bmatrix} = \boldsymbol{0} \tag{2.26}$$

for the variables $\boldsymbol{v}_{k,i} \in \mathbb{R}^{n_x}$ with $i \in [0, d]$, where $\boldsymbol{c}_k$ are the collocation conditions.

Moreover, a quadrature formula must approximate the cost function integral (2.24a), discretized as $\int_{t\ k}^{t_{k+1}} L(\boldsymbol{x}, \boldsymbol{u})dt$, on each collocation interval using the same points. This is denoted as $l_k(\boldsymbol{v}_k, \boldsymbol{s}_k, \boldsymbol{q}_k)$. The direct collocation method results in an NLP which is rewritten as

$$
\begin{aligned}
\min_{v,s,u} \quad & \sum_{k=0}^{N-1} l_k(\boldsymbol{v}_k, \boldsymbol{s}_k, \boldsymbol{q}_k) \\
\text{subject to} \quad & \boldsymbol{s}_0 - \boldsymbol{x}_0 = 0 && \text{(fixed initial value)}, \\
& \boldsymbol{c}_k(\boldsymbol{v}_k, \boldsymbol{s}_k, \boldsymbol{q}_k) = 0, k \in [0, N-1] && \text{(collocation conditions)}, \\
& \boldsymbol{p}_k(t_{k+1}, \boldsymbol{v}_k) - \boldsymbol{s}_{k+1} = 0, k \in [0, N-1] && \text{(continuity conditions)}, \\
& \boldsymbol{h}(\boldsymbol{s}_k, \boldsymbol{q}_k) \leq 0, k \in [0, N-1] && \text{(path constraints)}.
\end{aligned}
$$

2.4.2 Model predictive control

A Model Predictive Control (MPC) is a flexible method for solving an optimization problem over a predetermined, finite time horizon. The method splits the process into N time steps. It optimizes the control inputs needed for the system to reach the desired endpoint as an open-loop process, based on the dynamic model. Thereafter, it only performs the first control action before it solves the optimization problem again, with the current state values as initial conditions. This results in a close-loop optimization problem, which is more robust to modeling errors [66].

In determining the time step and horizon for the MPC, several factors must be considered. A large time step might result in a coarse approximation of the system dynamics, potentially leading to inaccurate model predictions. On the other hand, a small time step can yield more precise predictions, but this comes with the trade-off of increasing the computational time and complexity due to the increased number of calculations required.

The selection of the time horizon is also crucial. A short time horizon may not capture the long-term behavior of the system and might lead to myopic control decisions. Conversely, an excessively long time horizon could unnecessarily increase the computational burden and may include irrelevant future information.

The choice of these parameters should strike a balance between computational efficiency and predictive accuracy. This is typically achieved through a process of trial and error, guided by system-specific knowledge and requirements.

3

Methodology

The following chapter gives a presentation of how the theory is implemented to solve the docking problem. The first section offers a comprehensive explanation of the steps taken to prevent collisions. This includes defining the harbor and obstacles and elucidating their integration into the OCP. Furthermore, it describes the measures taken to ensure consideration of COLREGs. Moving forward, the chapter describes the algorithms employed for determining the shortest and smoothest path from the starting point to the docking point. The subsequent section focuses on explaining how the OCP is reformulated to represent the dynamic system. Lastly, an introduction is given to the software utilized for numerically solving the OCP.

3.1 Collision avoidance

To safely maneuver the vessel within a harbor, it is vital to have a good representation of the harbor and possible obstacles. Obstacles situated within the harbor area need to be taken into account and represented as outside the safe maneuverability area. Martinsen et al. (2019) provides a method for representing the harbor as spatial constraints which can be implemented in the OCP. Moreover, Bitar et al. (2019) and Ødven et al. (2022) offer a way of representing obstacles and generating safe waypoints by utilizing elliptic constraints and triangulation. This thesis utilizes a smooth path to warm-start the OCP. The implementation of these methods while taking into account the COLREG rules representation of the harbor, inspired by Eriksen et al. (2020), is explained in this section.

3.1.1 Harbor representation

A convex polygon $\mathbb{S}_s$ is defined as a representation of the harbor by choosing the largest convex area that encloses the desired endpoint. The convex polygons $\mathbb{S}_v$ and $\mathbb{S}_b$ are given as a representation of the vessel and a safety boundary enclosing the vessel, respectively.

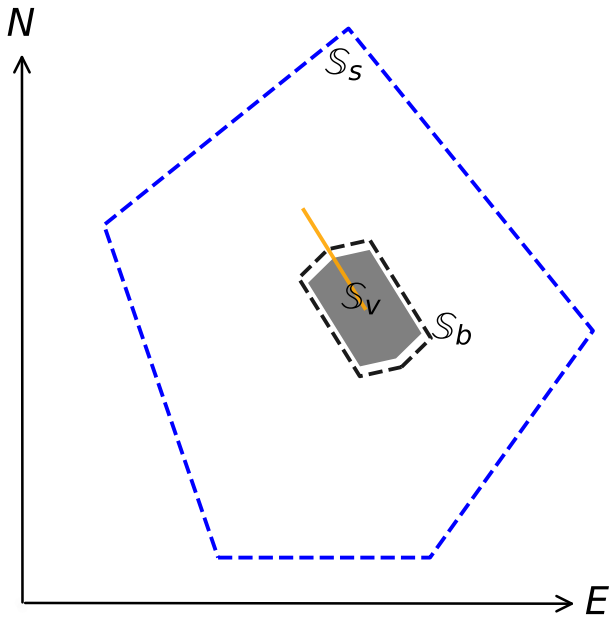


Figure 3.1: Illustration of the vessel $\mathbb{S}_v$ with black dashed lines as safety boundaries, $\mathbb{S}_b$, and the spatial constraints $\mathbb{S}_s$ represented by a blue dashed line. The orange line shows the heading of the vessel.

The illustration of these convex polygons is illustrated in Figure 3.1 and defined as:

$$\mathbb{S}_s = \{\forall\, \boldsymbol{v}_i^{NED} \in \text{Vertex}(\mathbb{S}_b) \mid \boldsymbol{A}_s \boldsymbol{v}_i^{NED} \leq \boldsymbol{b}_s\}, \tag{3.1}$$

$$\mathbb{S}_b = \text{Conv}(\mathbb{S}_v \oplus \mathbb{M}), \quad \mathbb{S}_b \subseteq \mathbb{S}_v. \tag{3.2}$$

$\mathbb{M}$ is the safety margin surrounding the vessel, $\boldsymbol{A}_s \in \mathbb{R}^{i\times 3}, \boldsymbol{b}_s \in \mathbb{R}^{i\times 1}$ and $\boldsymbol{v}_i^{NED} \in \mathbb{R}^2$, where i is the number of vertices. Since the vertices are represented in the NED reference frame, they must be transformed into the BODY frame to represent it as path constraints in the OCP. Consequently, the inequality constraints ensuring safe passage within the harbor is given by

$$\boldsymbol{A}_s \left(\boldsymbol{R}(\psi)\boldsymbol{v}_i^b + \begin{bmatrix} x \\ y \end{bmatrix} \right) \leq \boldsymbol{b}_s \;\forall\; \boldsymbol{v}_i^b \in \text{Vertex}(\mathbb{S}_b), \tag{3.3}$$

where $[x, y]^\top$ is the vessel position represented as Cartesian coordinates. ψ is the heading angle that is used to transform the kinematics and kinetics from the BODY frame to the NED frame with the rotation matrix

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix}.$$

Figure 2.4 illustrates the vessel heading and position relative to the NED frame.

3.1.2 Automatic convex harbor generation

This thesis aims to perform simulations in realistic, confined archipelagic areas. Such areas are often non-convex. Section 2.2.2 explains a method used in Martinsen (2021) in order to generate feasible convex harbors that can be implemented in the optimization problem as inequality constraints. Even though the method is efficient, it is comprehensive to implement. The main scope of this thesis is not to find the optimal method for convex set inner approximation and utilizes therefore a simple and sufficient approximation of the method. The implemented method uses the Python library, Shapely [67], in order to simplify the convex set generation process. On the other hand, the same concepts as in Martinsen (2021) and Deits and Tedrake (2015) are applied.

The Shapely library offers many intuitive functions for geometric manipulation and analysis. This makes finding the convex inner approximation trivial by following the steps:

1. Grow an ellipse from a center point $\boldsymbol{p}_c$ along the desired path until it intersects with an environmental constraint.
2. Find the tangent line of the intersection.
3. Grow an ellipse in one direction until it intersects with an environmental constraint.
4. Find the closest point, $\boldsymbol{p}_{ab}$, of intersection between the point $\boldsymbol{p}_c$ and the intersecting environmental constraint.
5. Find the normal line from $\boldsymbol{p}_c$ to $\boldsymbol{p}_{ab}$.
6. Repeat the process of growing an ellipse in desired directions and obtain the linear constraints.
7. Finding the convex inner approximation by splitting the non-convex harbor with the tangent lines and keeping the polygon containing the vessel.

In addition, this method provides a flexible implementation that simplifies the inclusion of the next waypoint when identifying the convex harbor. To ensure convexity when incorporating the next waypoint into the convex harbor, the algorithm can be initialized by creating an ellipse between the points (if possible), thus widening it until it intersects with an environmental constraint [40]. Increasing the feasibility and robustness of the system naturally involves including as many details about future movements as possible.

3.1.3 Obstacle representation

Triangulation

The convex set $\mathbb{S}_s$ represents the largest convex set within a harbor area that does not contain any landmass. It may therefore contain obstacles that a vessel must avoid. The triangulation presented in Section 2.2.3, offers a simple way of finding a set of points at an equal distance from the generating points. This method is used to find safe waypoints for the vessel by representing the generating points as the center of the obstacle. The waypoints are represented as the Voronoi edges' vertices, shown in Figure 3.2. To generate

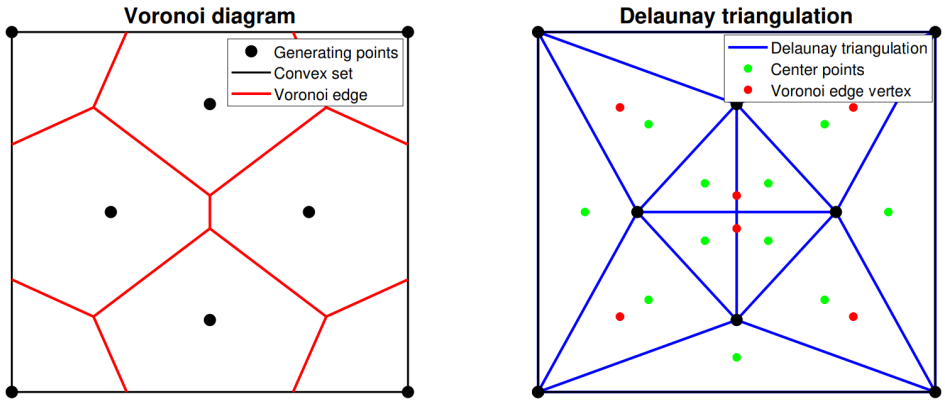


Figure 3.2: Constrained Delaunay triangulation and Voronoi diagram.

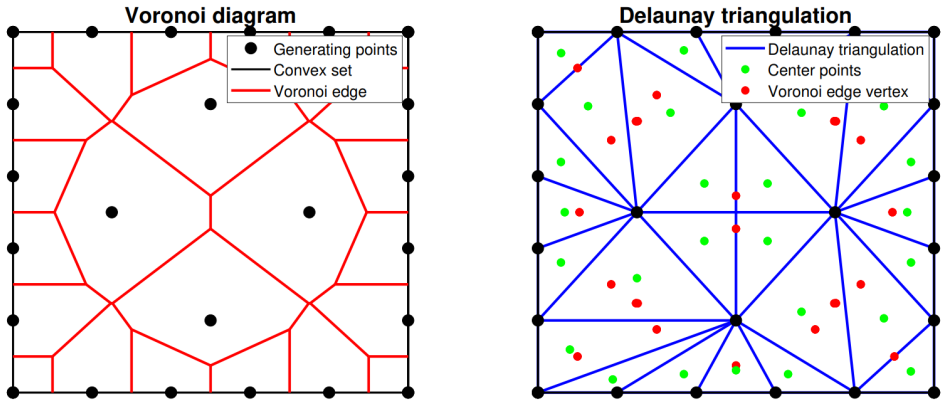


Figure 3.3: Constrained Delaunay triangulation and Voronoi diagram with added vertices along the convex set.

points that are at an equally safe distance from the harbor and the obstacles, the vertices defining the convex harbor region can be used to generate a constrained triangulation.

If the number of obstacles in a harbor area is increased, scenarios may occur whereas the few waypoints generated are not possible to traverse to without colliding with an obstacle. Therefore, increasing the number of vertices representing the harbor will simultaneously increase the waypoints with equal distance between the harbor and each obstacle. Figure 3.2 and 3.3 illustrates how the triangulation differs with an increase of vertices in the convex set $\mathbb{S}_s$. Moreover, to increase the number of possible waypoints between each obstacle, the waypoints can also be defined as the center of each Delaunay triangle. Consequently, the convex set $\mathbb{S}_s$ could contain several waypoints avoiding the obstacles. It is important to note, as mentioned in Section 2.3.3, that the worst-case computational complexity of

the A* search increases exponentially with the depth of the search tree. Therefore, the desired amount of generated waypoints must be chosen carefully in order to ensure a readily search algorithm. This thesis will therefore only utilize the Voronoi diagram and resort to Delaunay triangulation combined with Voronoi diagram only when no feasible path is found.

Ellipse constraints

The triangulation does offer waypoints that can be used when planning a collision-free path. On the other hand, it does not give the OCP any information as to where it can and cannot maneuver. Therefore, it's vital to incorporate these as constraints in the OCP. In this thesis, the obstacles are symmetric polygons with a given width and length. In order not to maneuver too close to the obstacles, they are represented by a region of collision (ROC) as elliptic inequalities that can be encoded as constraints in the OCP. An ellipse is given by the function

$$\left(\frac{x - x_c}{x_a}\right)^2 + \left(\frac{y - y_c}{y_a}\right)^2 \geq 1, \tag{3.4}$$

where x_c and y_c are the centre of the ellipse, and x_a and y_a are elliptic axes. From this, the inequality constraints that are encoded in the OCP are described similarly as in Ødven et al. (2022), inspired by Bitar et al. (2019),

$$\begin{aligned} &- \log\left[\left(\frac{(\boldsymbol{v}_{x,i} - o_{x,j})\cos(\alpha) + (\boldsymbol{v}_{y,i} - o_{y,j})\sin(\alpha)}{x_{a,j}}\right)^2\right. \\ &\left. + \left(\frac{(\boldsymbol{v}_{y,i} - o_{y,j})\sin(\alpha) + (\boldsymbol{v}_{y,i} - o_{y,j})\cos(\alpha)}{y_{a,j}}\right)^2 + \epsilon\right] \\ &+ \log(1 + \epsilon) \leq 0, \;\; \forall \boldsymbol{v}_i^b \in \text{Vertex}(\mathbb{S}_b), \end{aligned} \tag{3.5}$$

where $o_{x,j}$ and $o_{y,j}$ are the center of the obstacle j in Cartesian coordinates, where $j \in [1, \ldots, n]$ and n is the number of obstacles. The elliptic axes for each obstacle are given by $y_{a,j}$ and $x_{a,j}$. In order to avoid singularity when $\boldsymbol{v}_{x,i} \to o_{x,j}$ and $\boldsymbol{v}_{y,i} \to o_{y,j}$, a small constant $\epsilon > 0$ is added.

The uncertainty around the positioning of dynamic obstacles is typically larger than static obstacles. Additionally, the COLREGs state that the vessels should keep out of the way of each other. The region of collision (ROC) is, therefore, larger for dynamic obstacles and dependent on their velocity. The implementation of and differentiation between dynamic and static obstacles are illustrated in Figure 3.4.

3.1.4 COLREGs compliance

To ensure that the vessel follows the COLREG rules (Table 2.2) it is important to have a clear definition of how they are implemented when solving the docking problem. Complying with COLREGs involves two steps. The first step is to identify the rules that are relevant to the situation, followed by executing the appropriate course of action based on

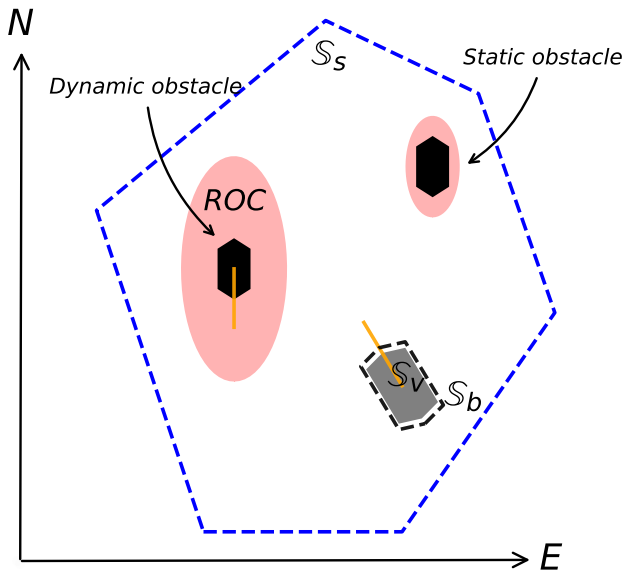


Figure 3.4: Illustration of the vessel $\mathbb{S}_v$ with black dashed lines as safety boundaries, $\mathbb{S}_b$, and the spatial constraints $\mathbb{S}_s$ represented by a blue dashed line. A static and a dynamic obstacle are also illustrated as black polygons with respective regions of collision (ROC) shown in transparent red. The orange lines show the heading of the vessel and the dynamic obstacle.

those rules [43]. Once the situation is classified this thesis utilizes the COLREGs by generating a forbidden area respective to each rule. Thereby, implementing these forbidden areas in the search algorithm. The sizes of the forbidden areas are also dependent on the speed of the dynamic obstacle in order to relieve unnecessary occupied space. An illustration of the COLREGs interpretation is shown in Figure 3.5. Figures 3.6 and 3.7 illustrate how penalized areas are implemented in the search algorithm based on the COLREGs interpretation.

The COLREGs provide situations, such as head-on encounters, where the OS must adjust its course to starboard and overtake the TS on its port side. Therefore, in such scenarios, the forbidden areas include an extra penalty in order to favor this maneuver, as shown in Figure 3.7. It is important to note that the COLREG rules only apply to one-on-one encounters between vessels. This thesis will consider several vessels, each independently with regards to COLREGs, therefore the proposed method is merely COLREGs inspired.

Classification

In order to classify which COLREGs are applied to the OS, two assumptions have been made:

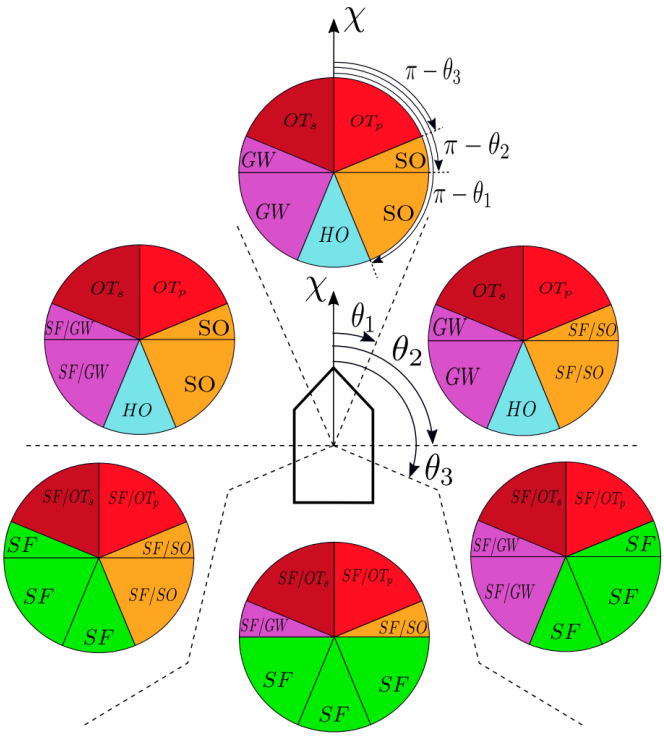


Figure 3.5: Illustration of COLREG rules 13-17 interpretation from Eriksen et al. (2020).

- **Assumption 1:** *All details regarding the TS's pose are known to the OS.*
- **Assumption 2:** *During an encounter, all target ships keep constant course and speed* [43].

Although the first assumption may not be a realistically feasible assumption, it is a necessary assumption in order to test the efficiency and robustness of the proposed method. On the other hand, technology such as automatic radar plotting aid (ARPA) and light detection and ranging (LIDAR) have shown to be effective at estimating moving obstacles [68, 69, 70]. The second assumption is reasonable in overtaking and give-way scenarios as the COLREG rules oblige the TS to keep a constant speed and heading to comply with rule 17.

To adhere to the conduct of vessels in any condition of visibility, assumption 1 along with CPA is used to cover rules 5 and 7. Rules 6 and 8 are taken into account with putting the OS in different maneuverable states as described in Section 3.1.5. Thereby, weighting the cost matrices and computing the desired speed along the path with equation (2.20), accordingly. Rule 9 is implemented by adding a small penalty to the possible waypoints on the port side of the harbor when the OS is faced toward the docking point. As for adhering to the conduct of vessels in sight of one another, rules 13-17, scenario classification is vital.

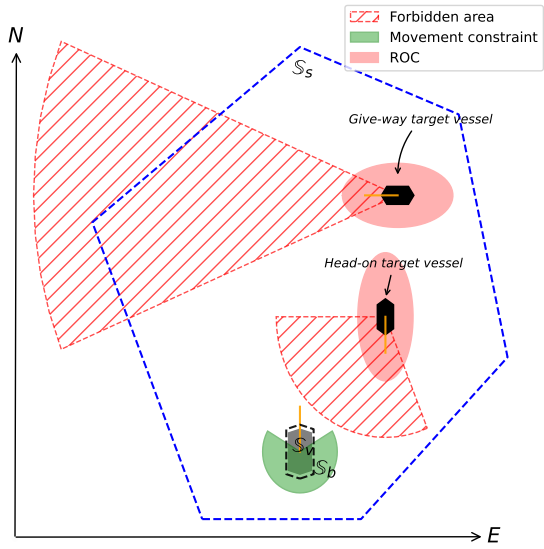


Figure 3.6: Illustration of the vessel and harbor, similar to Figure 3.4, where the transparent green is the vessel's movement constraint. Two TS' are shown, one in a head-on scenario and one in a give-way scenario. The TS' are illustrated with their penalized area, which is implemented in the search algorithm, according to the COLREGs.

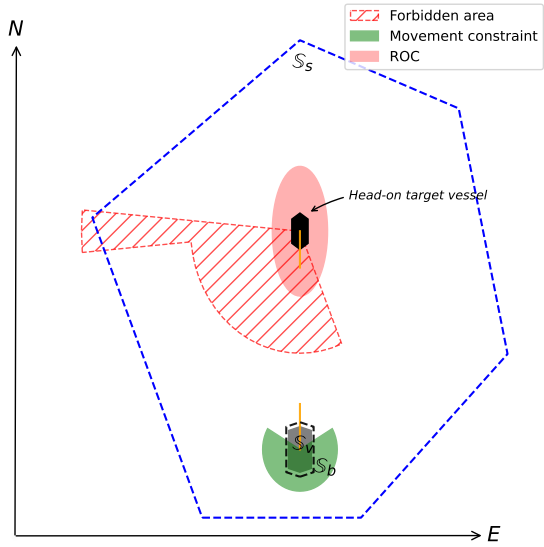


Figure 3.7: An illustration of the vessel $\mathbb{S}_v$ and a dynamic obstacle, similar to Figure 3.6. This figure shows how an additional penalty is added to the forbidden area in a scenario where the vessel must pass the TS on its port side.

In Tam and Bucknall (2010) and Eriksen et al. (2020), the COLREGs are interpreted by evaluating the relative bearing of the TS to the OS by computing

$$\phi = \arctan\left(\frac{y_{TS} - y_{OS}}{x_{TS} - x_{OS}}\right) - \chi_{OS}, \tag{3.6}$$

where χ_{OS} is the OS's course, and $p_{TS} = [y_{TS}, x_{TS}]$ and $p_{OS} = [y_{OS}, x_{OS}]$ are the position to the TS and OS, respectively. The relative bearing along with the TS's heading relative to the OS's heading, $\chi_{rel} = \chi_{TS} - \chi_{OS}$, classifies the situation as one of the following [11]:

- Overtaking (OT)
- Head-on (HO)
- Give-way (GW)
- Stand-on (SO)
- Safe (SF)

Similarly to Eriksen et al. (2020), the overtaking scenario is denoted into two different scenarios, OT_s and OT_p. These two different scenarios determine whether the OS should pass the TS on the starboard or port side, respectively. Overtaking on the port side is chosen if $\chi_{rel} \geq 0$ and starboard if $\chi_{rel} < 0$. The different scenarios and their geometrical interpretation are illustrated in Figure 3.5. The different sectors are given by the angles $[\theta_1, \theta_2, \theta_3] = [22.5^o, 90^o, 112.5^o]$ relative to the OS's course. Moreover, the circles with different situations, represent the TS in each relative bearing sector. In sectors containing two different scenarios, the former is chosen if the TS travels at a greater speed than the OS, and the latter if it is equal or smaller.

COLREGs and CPA

The COLREGs establish a set of rules that must be followed by all seafarers. However, within a harbor where the speeds typically are restricted to 5 $knots$, certain scenarios may occur where following the COLREGs does not render an acceptable maneuver. For instance, if the COLREGs evaluation result in an advice to overtake the TS, it may need to increase its speed to get safely ahead of the TS before docking. Therefore, utilizes this thesis TCPA and DCPA (2.7) to also verify if the COLREGs advice is feasible. If not, the vessel gives way to the TS.

3.1.5 Path planning

The vessel's maneuverable states

Maneuvering through a congested harbor will lead to different scenarios, desiring different behaviors. In order to make a more robust system, the scenario is evaluated and the vessel is given a state. These states are used to help weight the different cost functions formulated in section 3.2. The different maneuverable states have been chosen as:

- **Transit:** the COLREGs consider the scenario as safe.
- **Crossing:** the COLREGs consider the scenario anything other than safe.
- **Docking:** the vessel is in proximity to the desired end state.
- **Wait:** there is no collision-free course that exists from the initial to the desired state.

For a vessel to be considered successfully docked, it must fulfill three criteria: zero speed, correct heading, and correct position. While the vessel is in the docking state, it constantly checks which of these criteria it has not fulfilled, and weights the cost function accordingly. Moreover, the docking phase is initialized when the vessel is in proximity to the desired end point. In this thesis, that is within four times the vessel's length.

Shortest path

The Voronoi diagram generates a set of waypoints that avoids any obstacles. If a collision-free course exists from the initial to desired state, the A* algorithm is guaranteed to find the shortest one by traversing through the waypoints. The algorithm finds the shortest piecewise linear path by exploring the neighboring nodes and using heuristics to calculate the cost and guide its search [72]. In Bitar et al. (2019), the map is split into a uniform grid with subsequent drawbacks described in Section 2.2. To address some of these issues, Ødven et al. (2022) utilized the connected medial axes points to find the shortest path. One drawback with the method presented in Ødven et al. (2022), was that the algorithm and shortest path did not take the vessel's hull into account.

This thesis utilizes an adjusted A* algorithm that considers the COLREGs, and the vessel's heading and hull. The COLREGs are included in the algorithm as a penalized area in which the shortest path can not intersect. Similarly, the vessel's heading is taken into account by generating a speed-dependent, movement constraint around the vessel which the shortest path can not cross either, both illustrated in Figure 3.5. The vessel's hull is considered by checking if maneuvering toward the neighboring node will cause a collision. However, the implementation of forbidden areas in the search algorithm considerably restricts the search process. To address this issue, waypoints based on the vessel's predicted movement are included in order to increase the chances of finding a collision-free course.

Additionally, *gray areas* can be utilized to represent the space left by an obstacle, providing opportunities for later maneuvers. A set of waypoints are generated within the gray areas to help find a feasible path. If a path is found by utilizing the gray areas, the first point within that area is chosen as a temporary endpoint, namely *waitpoint*. This maneuver generates a human-like behavior and will help the vessel continue to move toward the desired endpoint and avoid unnecessary stopping. In case there is no feasible path to the desired state, a temporary endpoint, namely *waitpoint*, is set to a reachable waypoint with the lowest cost, and the vessel is set in a waiting state. In case the waypoint with the lowest cost is the same as the initial waypoint, the temporary endpoint is set to be the closest reachable waypoint. This will keep the vessel moving and avoid local minima by exploring new paths. The pseudocode for the adjusted A* algorithm is shown in **Algorithm 2**.

Algorithm 2: Search algorithm

Initialize
- *waypoints* ← VoronoiDiagram(**harbor**, **obstacles**)
- *waypoints* ← ConnectPointsToVoronoi(*startNode, endNode*)
- Initialize an empty $open$ and $closed$ list
- $g(startNode) = 0$
- $h(startNode)$ = Distance(*startNode, endNode*)
- $f(startNode) = g(startNode) + h(startNode)$
- Append *startNode* to $open$
- *moveConstraint* ← GenerateMovementConstraint(**vessel**)
- *waypoints* ← ConnectPointsToVoronoi(*predictedPoints*)

if ***vessel****.COLREG* ***is not*** *Safe* **then**
- *forbiddenArea* ← GenerateForbiddenArea(**obstacles**)
- *waypoints* ← PenalizeWaypoints(*forbiddenArea*)
- $grayArea$ ← GenereateGrayArea(**obstacles**)

while $open \neq \varnothing$
- *currentNode* ← $node \in open$ with lowest cost $f(node)$
- **if** *currentNode* = *endNode* **then**
  - **return**
- $open$.pop(*currentNode*)
- $closed$.append(*currentNode*)
- **for** *every neighborNode of currentNode*
  - **if** *neighborNode* $\in closed$ **then**
    - **continue**
  - **if** *Collision(currentNode, neighborNode)* **then**
    - **continue**
  - *possiblePath* ← StraightLine(*currentNode, neighborNode*)
  - **if** *possiblePath intersects with forbiddenArea* ***or*** *moveConstraint* **then**
    - **continue**
  - *cost* ← g(currentNode) + Distance(*currentNode, neighborNode*)
  - **if** *neighborNode* $\in$ *open* ***and*** *cost* $<$ *g(neighborNode)* **then**
    - *open*.pop(*neighborNode*)
  - **if** *neighborNode* $\in$ *closed* ***and*** *cost* $<$ *g(neighborNode)* **then**
    - *closed*.pop(*neighborNode*)
  - **if** *neighborNode* $\notin$ *open* ***and*** *neighborNode* $\notin$ *closed* **then**
    - *open*.append(*neighborNode*)
    - *g(neighborNode)* = *cost*
    - *h(neighborNode)* = Distance(*currentNode, endNode*)
    - *f(neighborNode)* = *g(neighborNode)* + *h(neighborNode)*

```
waitNode ← startNode
f(waitNode) = ∞
backupNode ← closest feasible neighboring node to startNode
for every node ∈ closed do
    if Collision(waitNode, node) then
        continue
    possiblePath ← StraightLine(waitNode, node)
    if possiblePath intersects with forbiddenArea or moveConstraint then
        continue
    if f(node) < f(waitNode) then
        waitNode ← node
if waitNode = startNode then
    waitNode ← backupNode
vessel.state = WAIT
return Path to waitNode)
```

Algorithm 3: Waypoint-reduction algorithm.

```
1  i ← N_*
2  ℙ ← InitializePath(p_i^*)
   while i > 1 do
       for j = 1 to i − 1 do
           if ¬Collision(p_i^*, p_j^*) then
3              AddPoint(ℙ, p_j*)
4              i ← j
5              break
```

It is undesirable to have a path with waypoints close to each other, as this prolongs the docking procedure. The waypoint-reduction algorithm from Bitar et al. (2019), is therefore implemented to shorten the path further. Ultimately, the shortest path is found with as few waypoints as possible. The pseudocode for the waypoint-reduction algorithm is shown in **Algorithm 3** and further described in Bitar et al. (2019).

3.1.6 Path smoothing

Offering a smooth path for a vessel to follow is vital in order to keep the planned path kinetically feasible. As mentioned in Section 2.3.3, there exist several techniques for smoothing a tangential discontinuous path. This thesis compares three different smoothing techniques, whereas the Bézier curve and B-spline are two of them, and chooses the shortest one. However, these two methods can only account for obstacles by adding control

Algorithm 4: Recursive path-smoothing algorithm.

```
1  oldPath ← reducedPath_start
   Repeat
       SmoothPath (newPath← oldPath_start, oldPath, waypointIndex ← 2)
2          i ← waypointIndex
3          if oldPath_i = oldPath_end then
4              newPath ← AddPoint(oldPath_i)
5              return newPath
6          waypoint ← newPath_end
7          nextPathStretch ← StraightLine(oldPath_{i−1}, oldPath_i)
8          midpoint ← nextPathStretch.interpolate(0.5, normalized=True)
9          exp ← 2
           while Collision(waypoint, midpoint) or
               intersectsPenalizedArea(waypoint, midpoint) do
10             midpoint ← nextPathStretch.interpolate(0.5^exp, normalized =True)
11             exp ← exp + 1
12         centroid ← center of Triangle(waypoint, midpoint, oldPath_{i−1})
13         newPath ← AddPoint(centroid)
14         newPath ← AddPoint(midpoint)
15         return SmoothPath(newPath, oldPath, i + 1)
16     oldPath ← newPath
```

points in proximity to an obstacle, and the path must be verified as feasible by traversing the path and checking for a collision. This thesis makes use of a recursive path-smoothing algorithm, outlined in **Algorithm 4**. This algorithm guarantees a path that is both collision-free and shorter than the original path, while also maintaining curvature continuity. The algorithm is inspired by de Casteljau's recursive method.

The algorithm takes the reduced path from **Algorithm 3** and the starting point as input. Thereby, it finds the midpoint on the line segment from the next waypoint to the one after. As long as this point causes a collision with an obstacle or COLREGs forbidden area, a new midpoint is found in the middle of the next waypoint and the previous midpoint. In order to ensure a curved path, a triangle is generated between the current waypoint, the next waypoint, and the midpoint. Thereafter, the triangle centroid along with the midpoint is utilized to generate a new path. This procedure is executed for each of the waypoints given by the reduced path. Running this algorithm once does not ensure a smooth path as shown by the blue line in Figure 3.8. Repeating the procedure several times with the new path instead of the reduced path, however, will return a path with continuous curvature.

It is desirable to find a new path quickly if the current path indicates collision risk. Therefore, becomes the time complexity of generating a smooth path an important aspect of path smoothing. There are mainly three elements that prolong the computational time of

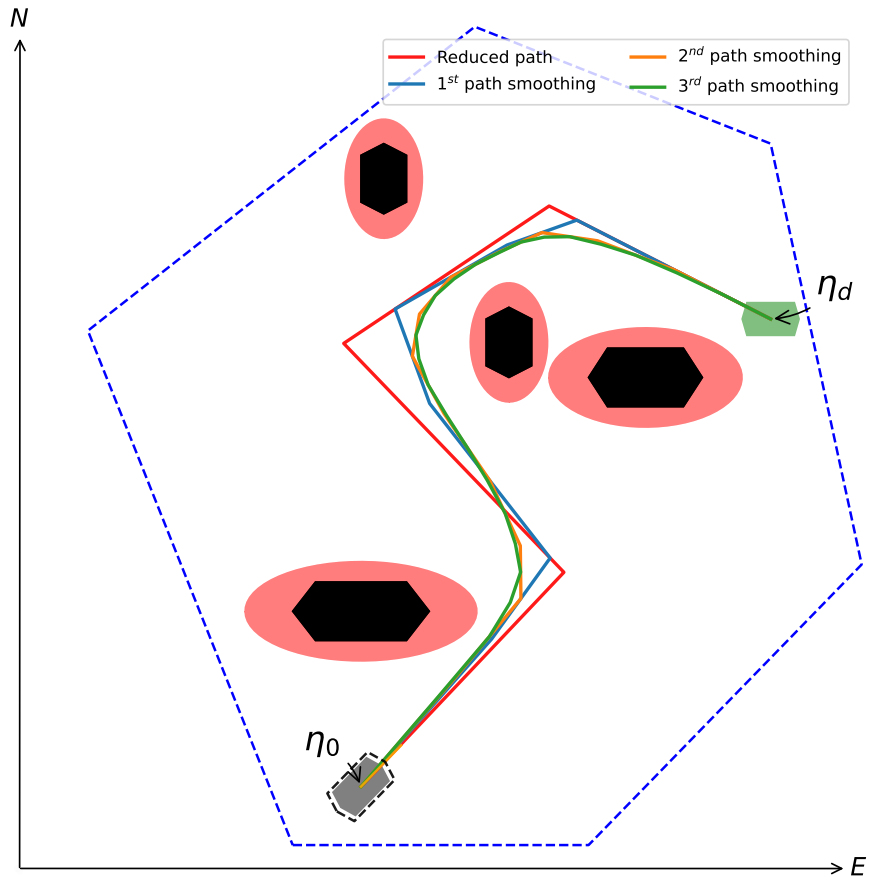


Figure 3.8: Illustration of the different collision-free paths generated by **Algorithm 4**.

Algorithm 4; how many times a new midpoint has to be calculated, the length of the proposed path, and finally how many times the whole procedure is repeated. The midpoint is typically not calculated no more than six or seven times, since it is always moving closer to a feasible waypoint, and at the waypoint when $0.5^{exp} \ll 1$. Therefore, the worst-case computational time of one recursion is $O(\log(d))$, where d is the distance between the two waypoints, indicating the number of times the line could be split to find a feasible point. The amount of waypoints in the proposed path determines how many times the recursive function will call itself, leading to a time complexity of $O(k^N)$, where $N = \log(d)$ is the time complexity of one recursion, and k is the number of waypoints in the proposed path. The algorithm's worst-case time complexity is $O(R \times k^{\log(d)})$, where R indicates the number of times the procedure is repeated. This computational complexity is not ideal, however, the algorithm's base-case complexity is expected to be $O(1)$ because of various factors. Firstly, the distance between waypoints in a harbor is typically short. Secondly, the number of waypoints is usually a small constant value. Finally, the amount of times the whole procedure is repeated is decided by the user, and as illustrated in Figure 3.8, two repetitions will render a sufficient path. Ultimately, leading to a fast computational time.

The smoothing algorithm also offers the possibility to adjust the desired curvature of the path by choosing the centroid point either as several points in a Dubin's curve or by a Fermat spiral, utilized in Candeloro et al. (2017) to generate a smooth path. However, utilizing the centroid point generates a sufficiently smooth and collision-free path and will therefore be used in this thesis.

Furthermore, this algorithm can be utilized to initiate a more feasible path for the Bézier curve and B-spline to follow. Figures 3.9 and 3.10 illustrate the resulting enhanced paths. The number of repetitions in **Algorithm 4** used for initiating the Bézier curve and B-spline depends on the desired amount of control points. In Figure 3.10, only one repetition is utilized and it is clear that this will generate curves that are close to the initial path. Evidently, the proposed path from **Algorithm 4** is the shortest feasible one, as the Bézier curve will make the vessel maneuver within the ROC.

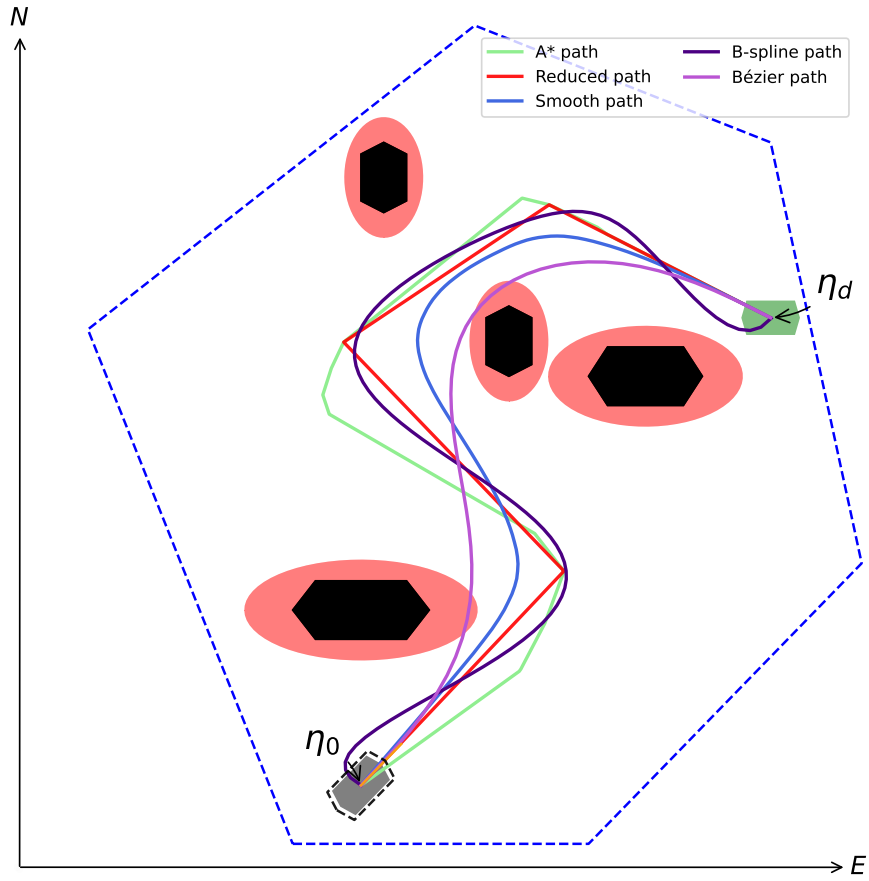


Figure 3.9: Shortest path from the initial state, η_0, to the desired state, η_d, with no added vertices along each side of the convex set. The black shapes are obstacles and the red ellipses surrounding them, are the elliptic constraints for the respective obstacle.

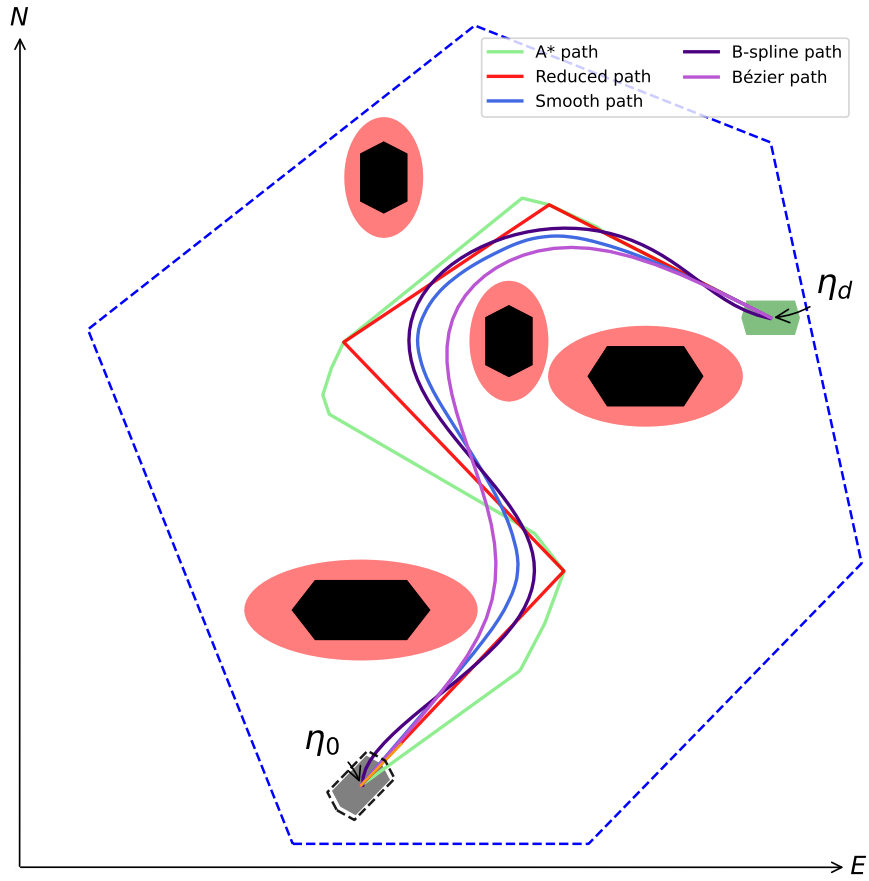


Figure 3.10: Shortest path from the initial state, η_0, to the desired state, η_d, with 10 added vertices along each side of the convex set. The black shapes are obstacles and the red ellipses surrounding them, are the elliptic constraints for the respective obstacle.

3.2 Optimal Control Problem

The OCP (2.24) can be reformulated by utilizing the vessel model and constraints presented in sections [2.1, 3.1]. The cost function, which will be minimized, consists of three components. The first component, c_η, is a simple quadratic cost function penalizing the positional error, velocity, input, and cross-track error, and is given by

$$c_\eta = ||\boldsymbol{\eta} - \boldsymbol{\eta_d}||^2_{\boldsymbol{Q_\eta}} + ||\boldsymbol{\nu}||^2_{\boldsymbol{Q_\nu}} + ||\boldsymbol{u}||^2_{\boldsymbol{R_u}} + ||y_e||^2_{y_e w} \tag{3.7}$$

where $\boldsymbol{\eta_d} = [x_d, y_d, \psi_d]^\top$ and $\boldsymbol{Q_\eta}, \boldsymbol{Q_\nu}$ and $\boldsymbol{R_u}$ are diagonal matrices used to weight the penalty of the respective states and control inputs. y_{ew} is the penalization of the cross-track error. The quadratic positional cost is only utilized when far away from the docking point. When in proximity to the docking pose, a Pseudo-Huber function is utilized to calculate the positional cost, similarly to Ødven et al. (2022). This function provides better results to the optimization problem as it improves the numerical stability (Gros and Diehl (2022)). The function is given by

$$c_{x,y} = \sigma\delta^2\left(\sqrt{1 + \frac{(x - x_d)^2 + (y - y_d)^2}{\delta} - 1}\right), \tag{3.8}$$

where σ and δ are a tunable variables.

The second component, c_ψ, adds a penalty on the vessel's heading. Similar to Martinsen et al., Ødven et al. (2020, 2022), the cost function is formulated as

$$c_\psi = \frac{1}{2}\big(1 - \cos(\psi - \psi_d)\big)e^{\frac{1}{2\delta^2}\left((x-x_d)^2 + (y-y_d)^2\right)} \tag{3.9}$$

A slack is added to the desired end state to reduce the need for tuning the penalty weight matrices and give the OCP more room to find an optimal solution. Resulting in the inequality constraints $-\boldsymbol{\epsilon} \leq \boldsymbol{\eta_d} - \boldsymbol{\eta}_{end} \leq \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} > \mathbf{0}$ is a small constant and $\boldsymbol{\eta}_{end}$ is the final state calculated by the OCP.

Ultimately, the OCP is given by:

$$\min_{\boldsymbol{\eta},\boldsymbol{\nu},\boldsymbol{f},\boldsymbol{\alpha}} \quad \int_0^T c_\eta + 20c_\psi \, dt \tag{3.10a}$$

$$\text{subject to} \quad \dot{\boldsymbol{\eta}} = \boldsymbol{J}(\psi)\boldsymbol{\nu}, \tag{3.10b}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{T}(\alpha)\boldsymbol{f}, \tag{3.10c}$$

$$\boldsymbol{A}_s\left(\boldsymbol{R}(\psi)\boldsymbol{v}_i^b + \begin{bmatrix} x \\ y \end{bmatrix}\right) \leq \boldsymbol{b}_s \; \forall \boldsymbol{v}_i^b \in \text{Vertex}(\mathbb{S}_b), \tag{3.10d}$$

$$-\log\left[\left(\frac{v_{x,i} - o_{x,j}}{x_{a,j}}\right)^2 + \left(\frac{v_{y,i} - o_{y,j}}{y_{a,j}}\right)^2 + \epsilon\right] + \log(1+\epsilon) \leq 0, \;\; \forall \boldsymbol{v}_i^b \in \text{Vertex}(\mathbb{S}_b), \tag{3.10e}$$

$$-\epsilon \leq \boldsymbol{\eta_d} - \boldsymbol{\eta}_{end} \leq \epsilon, \tag{3.10f}$$

$$\boldsymbol{f}_{min} \leq \boldsymbol{f} \leq \boldsymbol{f}_{max}, \tag{3.10g}$$

$$\boldsymbol{\alpha}_{min} \leq \boldsymbol{\alpha} \leq \boldsymbol{\alpha}_{max}, \tag{3.10h}$$

$$|\dot{\boldsymbol{\alpha}}| \leq \dot{\boldsymbol{\alpha}}_{max}, \tag{3.10i}$$

$$\text{Initial conditions on } \boldsymbol{\eta}, \boldsymbol{\nu}, \boldsymbol{f}, \boldsymbol{\alpha}, \tag{3.10j}$$

where the objective function $c_\eta + c_\psi$ is minimized over a time horizon $T > 0$. The state dynamics is formulated as (3.10b) and (3.10c). The path constraints (3.10d), (3.10e) and (3.10f), saturation constraints (3.10g), (3.10h), (3.10i) and initial conditions (3.10j) must also be satisfied. This thesis utilizes the same saturation constraints as in Martinsen et al. (2019), where the maximum thrust for the azimuth and tunnel thrusters are $\pm 1/30$ and $\pm 1/60$ of the dry ship's weight, respectively. The constraints on the azimuth angles (3.10h) are $\pm 170^o$ giving a forbidden area of 20^o such that they will not generate thrust in opposite directions, potentially damaging the thrusters. A maximum turnaround time for the azimuth thrusters is also added as $30s$ per revolution to give a realistic simulation of how azimuth thrusters move.

The OCP (3.10) is utilized as a fundament for solving a nonlinear MPC. If the OCP were only to be solved once, it would give an open-loop trajectory over the time horizon. The process is therefore divided into N time samples, where the OCP is solved for a finite time horizon with the vessel states used as initial conditions. Before the prediction of the following time step, only the first predicted control action is executed. Resulting in a closed-loop optimization problem, making the method more robust to modeling errors.

3.3 CasADi

CasADi is an open-source software tool, developed by Andersson et al. (2018), which is used to simplify the implementation of optimization problems. CasADi offers several different numerical solvers to solve an OCP. This thesis will utilize an interior point optimizer (IPOPT) [74] for solving the NLP given by a direct collocation transformation of the OCP (3.10). The implementation of the direct collocation method is a recreation inspired by Andersson's implementation in 2016 [75].

4

Simulation results and discussion

The following chapter presents and discusses the results obtained from solving the docking problem while adhering to COLREG rules. The optimal path is determined by traversing a Voronoi Diagram using **Algorithm 2**, reducing the number of waypoints with **Algorithm 3**, and subsequently smoothing the path with **Algorithm 4**. The determined optimal path ensures avoidance of static obstacles by checking for collision-free trajectories with respect to the vessel's hull. Furthermore, the path accounts for dynamic obstacles while adhering to COLREGs. This is achieved by calculating the TCPA and DCPA using 2.7, determining the relevant COLREG rule, and recalculating an optimal path accordingly. In addition, considering the non-convex nature of the maneuvering area, online generation of feasible convex areas is necessary. Convex sets are generated and stored along the initial optimal path to optimize computational efficiency. The simulations are performed on a Dell Optiplex 7090.

The simulations took place in the Kragerø archipelago, as shown in Figure 4.1. This harbor area is typically crowded with leisure boats during the summer, posing increasing challenges to the docking problem. The simulations are divided into two sections: COLREG scenarios and a complex scenario. The simulations in the first section demonstrate the vessel's adherence to COLREGs, while the second section showcases the system's behavior when faced with multiple obstacles in the harbor area.

The OCP was implemented as a closed-loop Nonlinear Model Predictive Control (NLMPC). To balance computational time and prediction accuracy, an adaptive time step and time horizon were employed. The time horizon was determined by the approximate time it would take for the OS to reach the LOS pose, given by

$$T_i = \min\big(T_{\text{max}}, \max\big(T_{\text{min}}, \frac{\Delta}{||\boldsymbol{v}_{OS}||}\big)\big), \tag{4.1}$$

at iteration i, where $T_{\text{min}} = 10s$ and $T_{\text{max}} = 30s$ were chosen as the minimum and maximum time horizon, respectively.

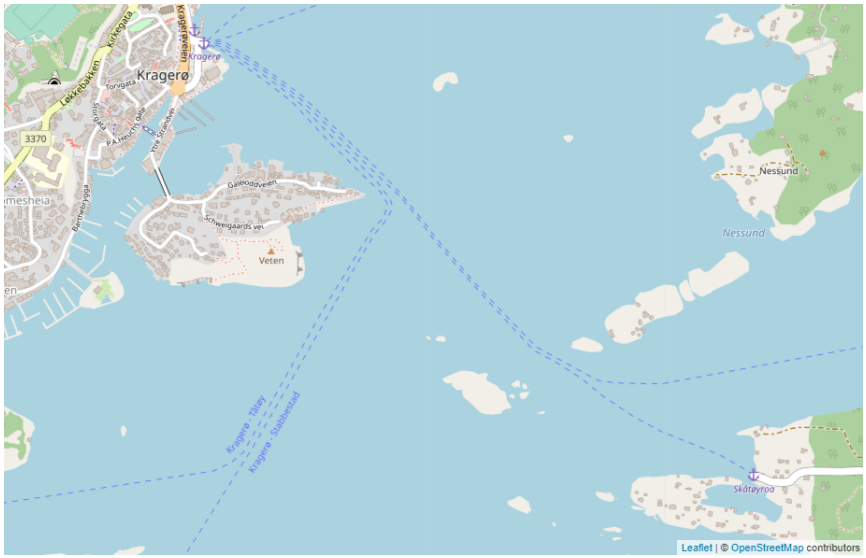


Figure 4.1: Illustration of the simulation area, the Kragerø archipelago. The blue dotted lines represent typical ferry crossings.

The number of time steps was determined to linearly increase with the cross-track error, as given by

$$N_i = \min(N_{\min} + y_e, N_{\max}) \tag{4.2}$$

at iteration i, with $N_{\min} = 10$ and $N_{\max} = 30$ as the chosen minimum and maximum time steps, respectively.

The ASV follows the optimal path using the LOS-guidance law (2.16) with an adaptive lookahead distance. The desired speed at each LOS point is determined using CB guidance (2.20), with a maximum approach speed equal to the ASV's operating speed. The milliAmpere's maximum speed is 5 $knots \approx 2.57\ m/s$, whilst it has an operating speed of 3 $knots \approx 1.54\ m/s$ [9].

4.1 COLREG scenarios

In this section, the ASV started at the Kragerø ferry terminal and docked at a jetty on Øya. The objective was to test COLREG rules 13-17. The optimal path was computed within 1.1 seconds, and new convex areas were generated in less than 0.06 seconds. The initial and desired positions were the same for all the following scenarios. Consequently, the relevant convex areas were loaded before initializing the OCP. The operating convex area was chosen to contain the vessel and as much of the optimal path as possible.

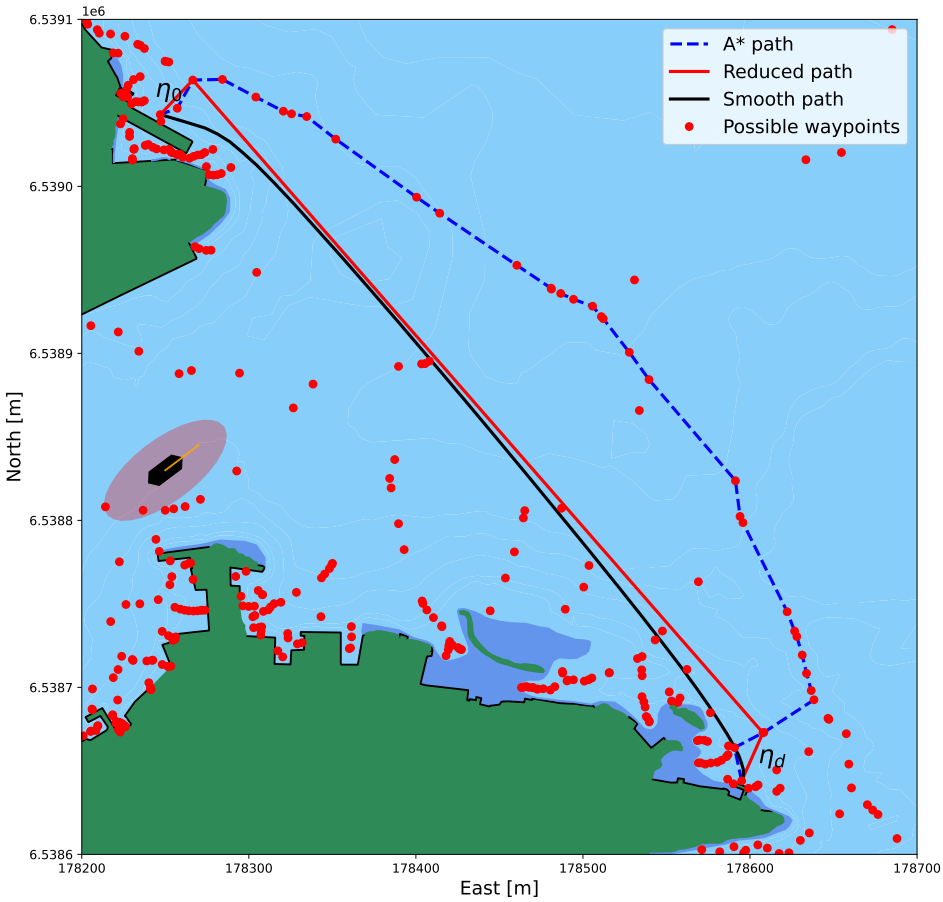


Figure 4.2: Optimal path from initial pose $\boldsymbol{\eta_0}$ to the desired pose $\boldsymbol{\eta_d}$.

4.1.1 Give-way scenario

The give-way scenario depicted a realistic situation where the ASV crossed the port exit/entrance while another vessel was exiting the harbor, requiring the ASV to give way and adhere to COLREGs. The TS was traveling at 2 m/s and had dimensions of 25 meters in length and 10 meters in width. When a collision risk was detected, a new optimal path was computed within 1.2 seconds. The ASV's trajectory and predicted movements before and after replanning were illustrated. Since the replanned path was significantly different from the initial path, no convex areas encompassed the entire trajectory. Therefore, new convex sets were computed in 0.06 seconds. Once the TS had passed and the scenario was considered safe, the optimal path was updated within 0.7 seconds.

The docking problem was successfully solved in the give-way scenario while maintaining a safe distance from the land and the dynamic obstacle, as shown in Figure 4.6. Moreover,

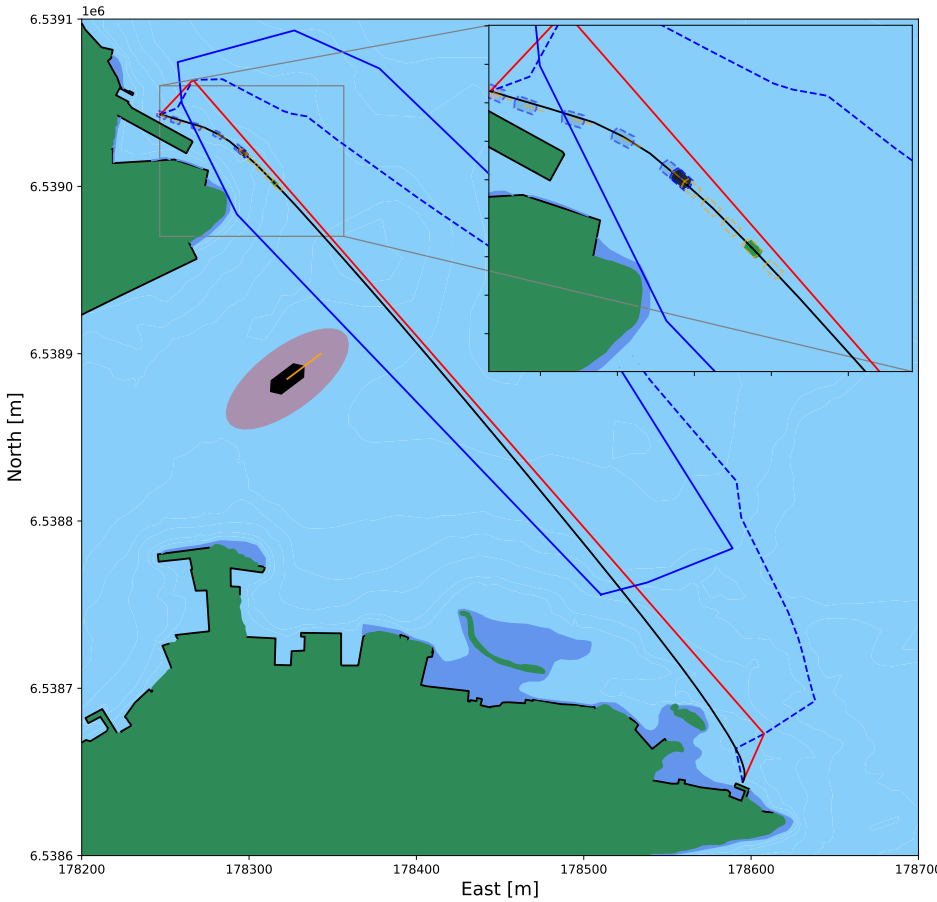


Figure 4.3: Trajectory of the give-way docking scenario, when the systems detect a collision risk. The current position is plotted in black, desired LOS-pose in green, future predictions in orange, and shadow ships in gray. The blue-lined convex set is the current safe operation area for the ASV.

the ASV took an early action that was large enough to be readily observable for the target ship. The total simulation took approximately 355 seconds while computing the entire scenario took 137 seconds. Rendering an average of 0.39 seconds to compute each iteration in the NLPMC. Since the system computed the next scenario faster than it executed it, it was realistic and feasible.

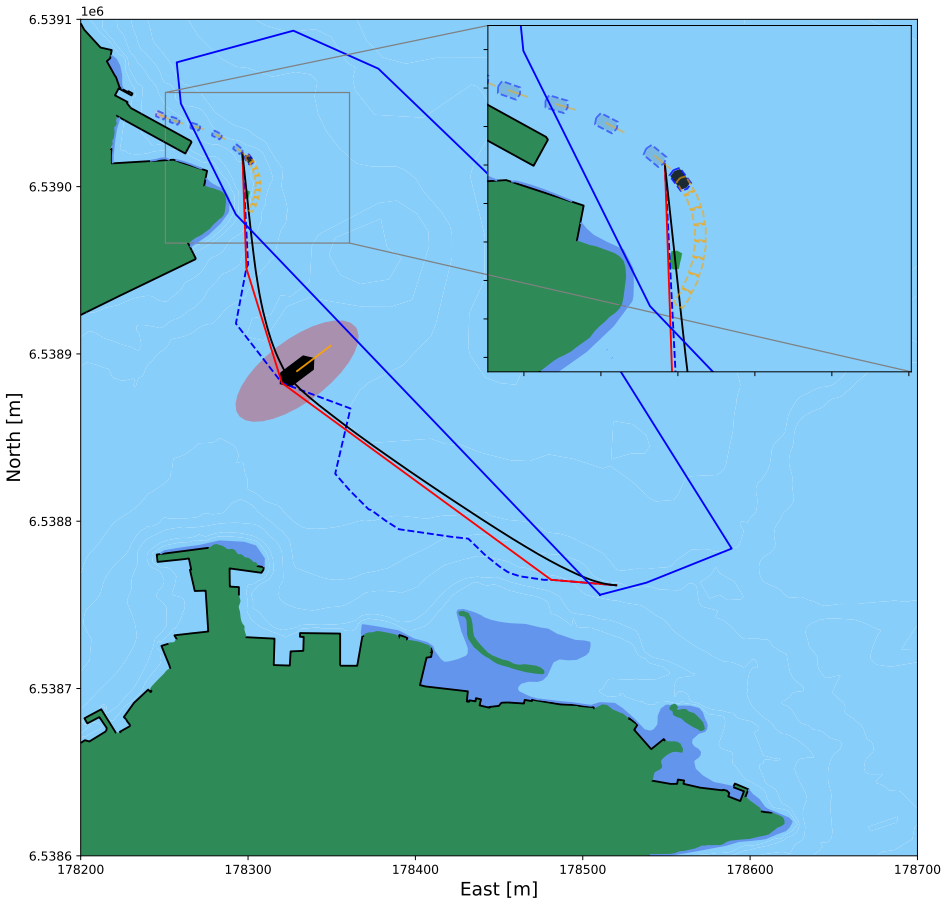


Figure 4.4: Illustration of the ASV's reaction to detecting a collision risk in a give-way scenario.

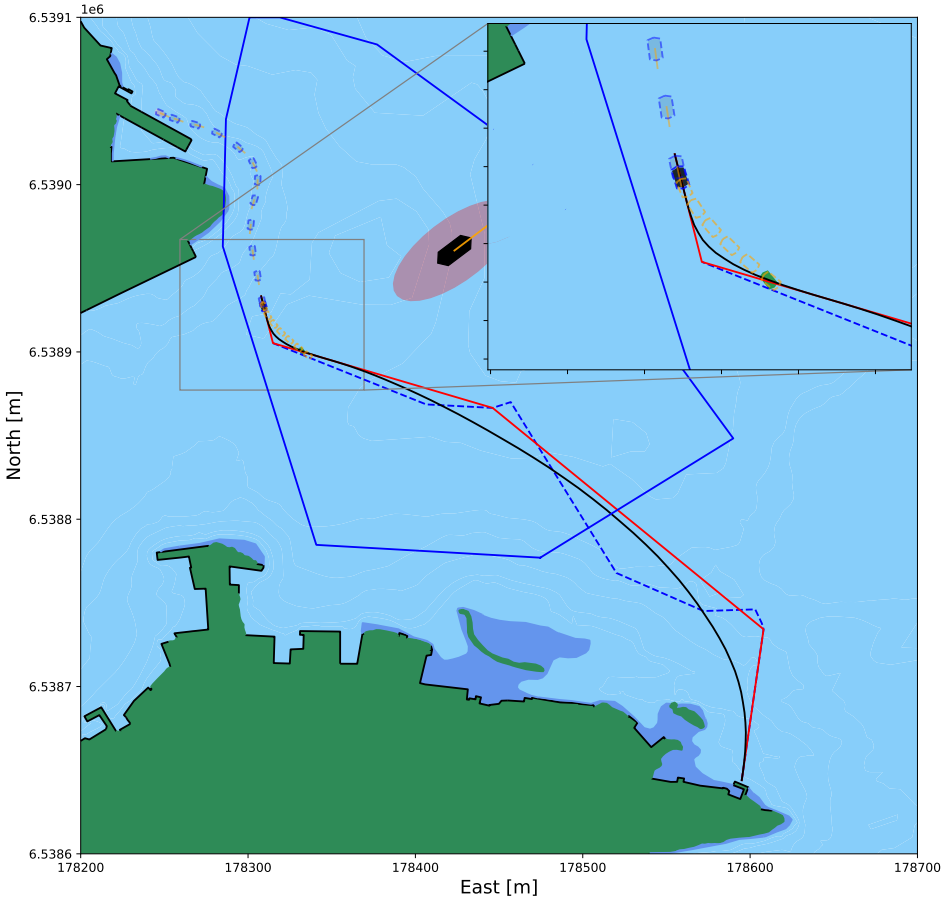


Figure 4.5: Illustration of the ASV updating the optimal path when the situation is considered safe.

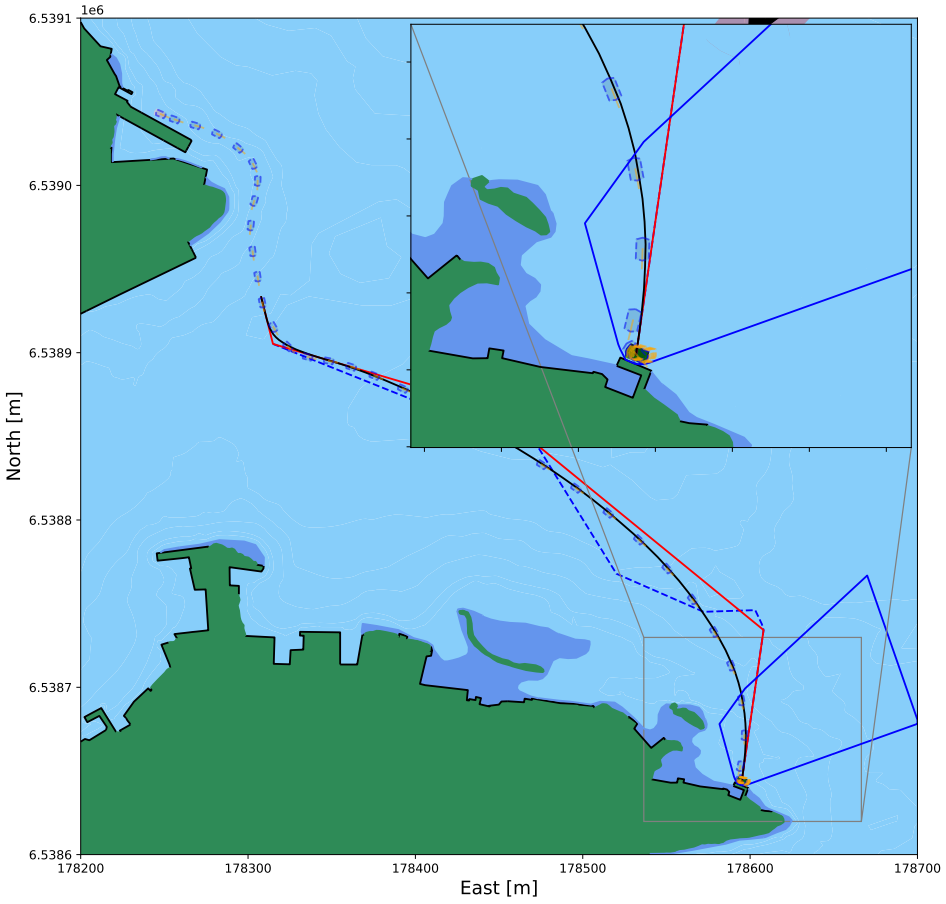


Figure 4.6: Complete trajectory of the give-way docking scenario, zoomed in on the final position. The final position is plotted in black, desired position in green, future predictions in orange, and shadow ships in gray.

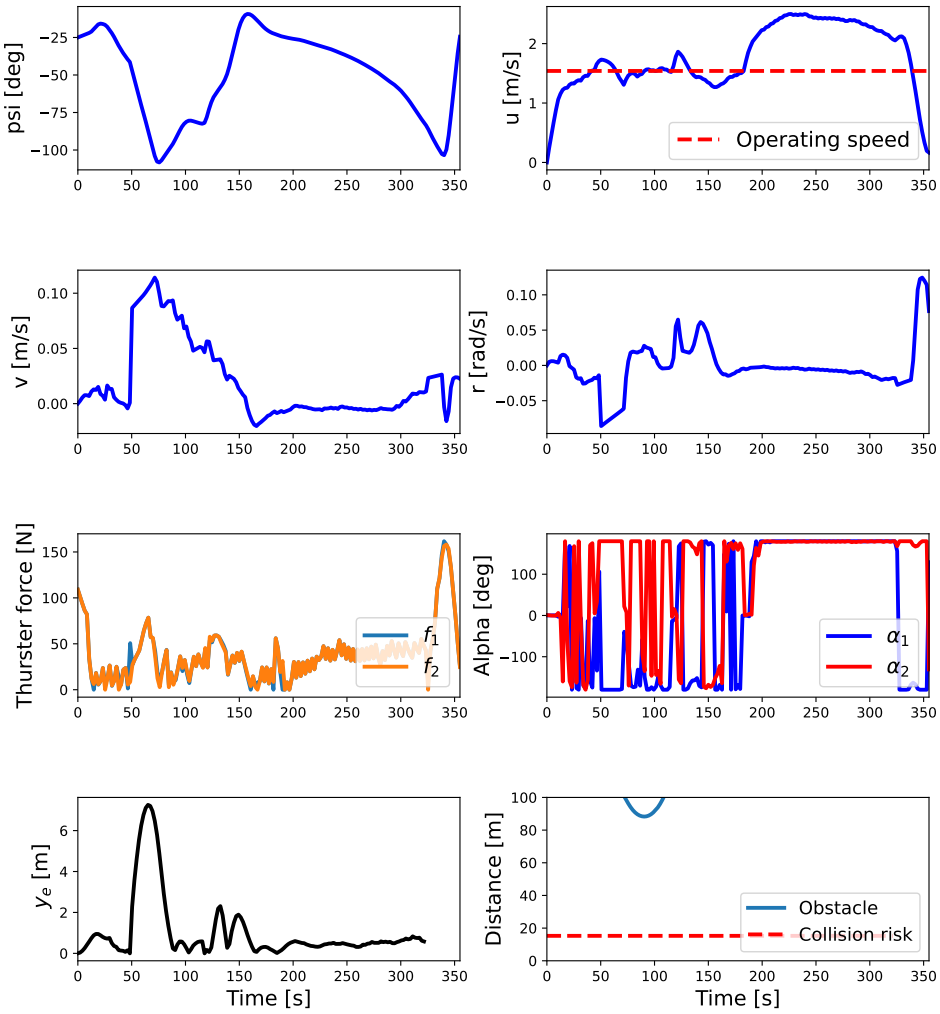


Figure 4.7: States $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$, along with cross-track error and distance from obstacle when crossing and docking in a give-way scenario.

The ASV's states, $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$, for the give-way scenario, are shown in Figure 4.7. Around 50 seconds into the simulation, the optimal path is updated, causing a spike in the cross-track error. The vessel maintained its surge speed around the operating speed until 200 seconds into the simulation when the scenario was deemed safe. It is evident that the ASV changed its heading before coming into close proximity to the obstacle, thereby adhering to the COLREGs. Although the azimuth angles' saturation constraints were satisfied, they exhibited more oscillation than would be ideally desirable. Examining the azimuth angles, yaw rate, and maneuver depicted in Figure 4.6, it becomes evident that the vessel effectively utilizes the azimuth features, executing a docking maneuver resembling human-like behavior.

4.1.2 Head-on scenario

Head-on scenarios pose a significant risk of collision, where both parties must alter their course to starboard to avoid a collision. However, there is no guarantee that the target ship will take the right action in time. The following simulation is therefore performed under similar assumptions as the rest, proposed in Section 3.1.4.

In this scenario, the TS is headed for the fuel station next to Kragerø terminal, which could potentially cause a head-on collision. The TS had the same constant speed and dimensions as in the give-way scenario. Figure 4.8 illustrates the moment when the OS detects a head-on collision risk. A new optimal path, considering the COLREGs, is found within 0.6 seconds and is shown in Figure 4.9. The replanning was initialized when the OS was relatively close to the TS. However, the OS was aware of the collision risk early on, but the replanning of the optimal path was triggered based on a tuning variable that could be adjusted. By keeping this variable relatively small, it illustrated the OS's ability to react quickly and avoid the TS.

Once the ASV considered the scenario safe, it updated the optimal path, which was found in less than 1 second, as shown in Figure 4.10. The updated path was found in less than 1 second. Similar to the give-way scenario, the total simulation time took about 355 seconds, where 70 of which were spent in the docking phase. It took a total of 151 seconds to compute the entire scenario, however, it took only 85.8 seconds to compute the trajectory up until the docking phase. Resulting in an average of 0.3 seconds to compute each iteration in the NLMPC before docking, and 0.93 seconds when docking. This was due to the vessel reaching a local minimum before successfully docking, as depicted in Figure 4.11. However, due to the different weighting of the objective function (Section 3.1.5), the ASV was able to converge to the desired endpoint. The suboptimal pose near the desired pose may have been caused by the relatively sharp turn preceding the docking phase. Nevertheless, the OS successfully solved the docking problem by passing the dynamic obstacle on its port side, complying with the COLREGs.

The ASV's states, including cross-track error and distance from the obstacle, are shown in Figure 4.13. The cross-track error was only measured until the vessel reaches the docking phase, starting approximately 285 seconds into the simulation. By not measuring the cross-track error after the vessel was in proximity to the final destination, the OCP was

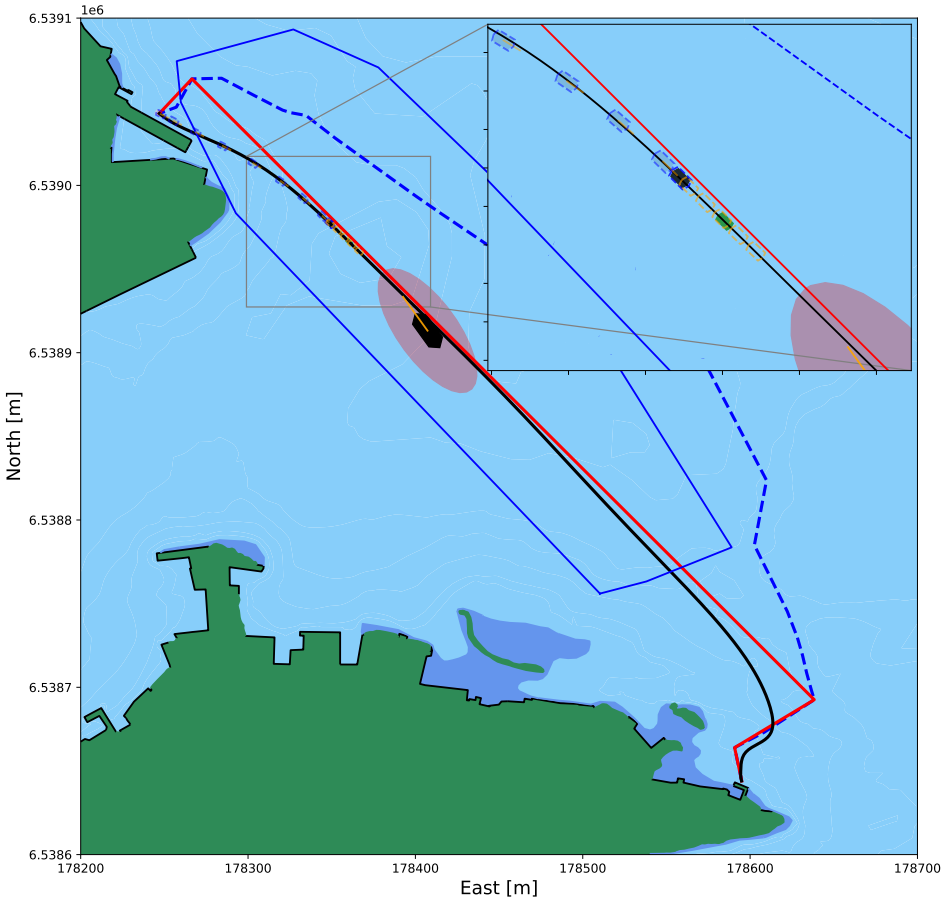


Figure 4.8: Trajectory of the give-way docking scenario, when the systems detect a collision risk. The current position is plotted in black, desired LOS-pose in green, future predictions in orange, and shadow ships in gray. The blue-lined convex set is the current safe operation area for the ASV.

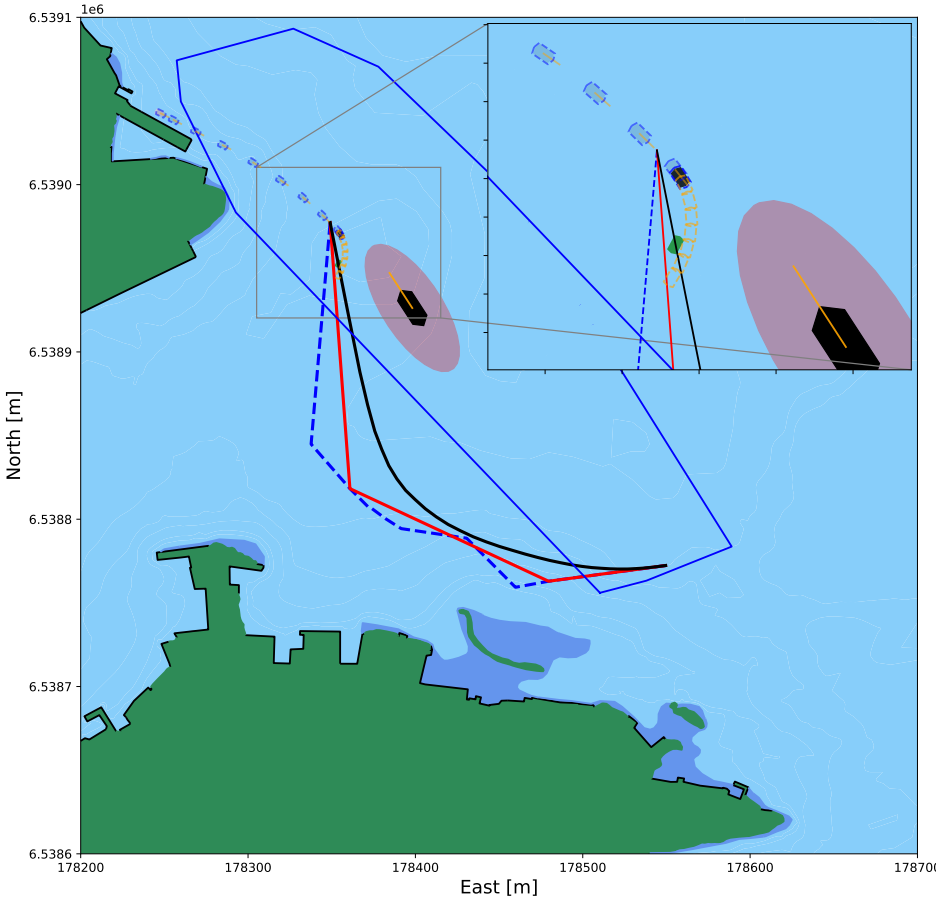


Figure 4.9: Illustration of the ASV's reaction to detecting a collision risk in a give-way scenario.

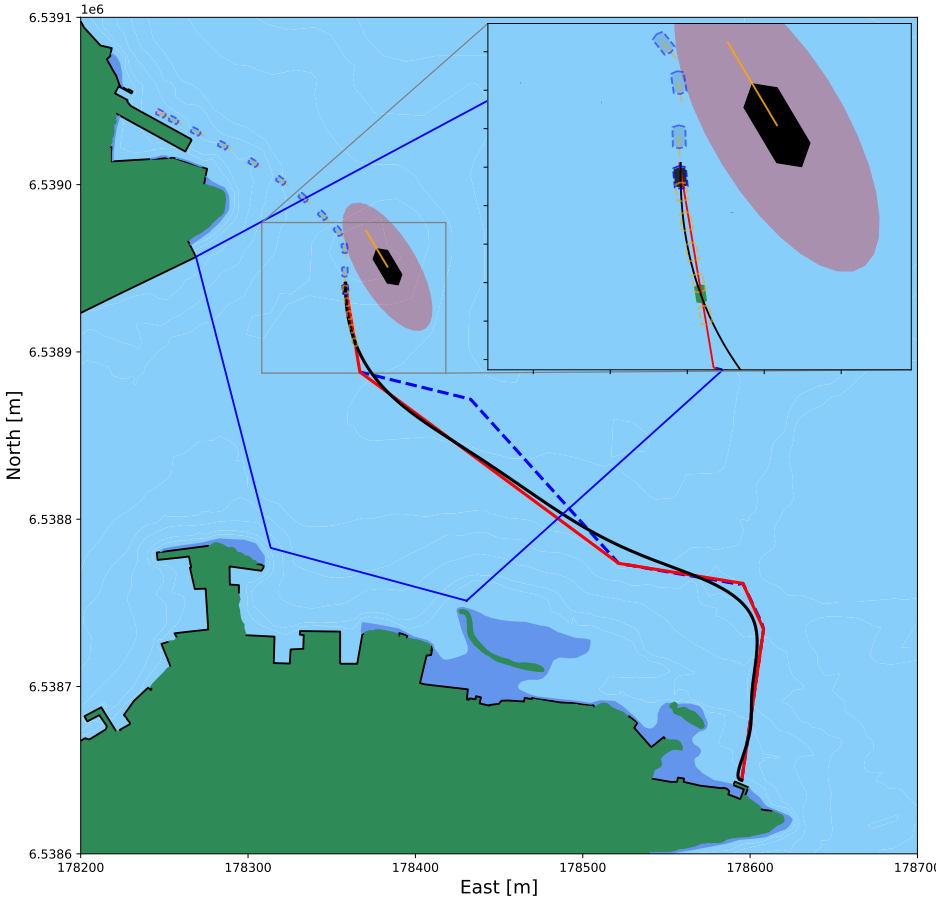


Figure 4.10: Illustration of the ASV updating the optimal path when the situation is considered safe.

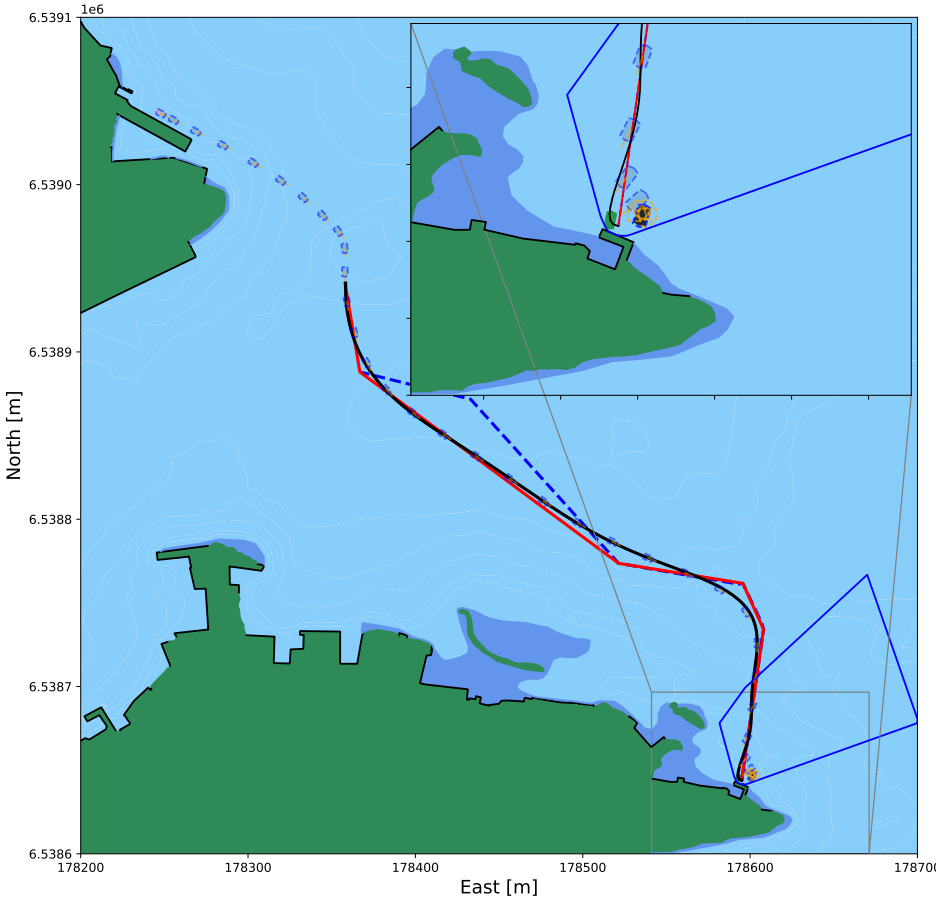


Figure 4.11: Illustration of the ASV reaching a local minimal in the docking phase of the simulation.

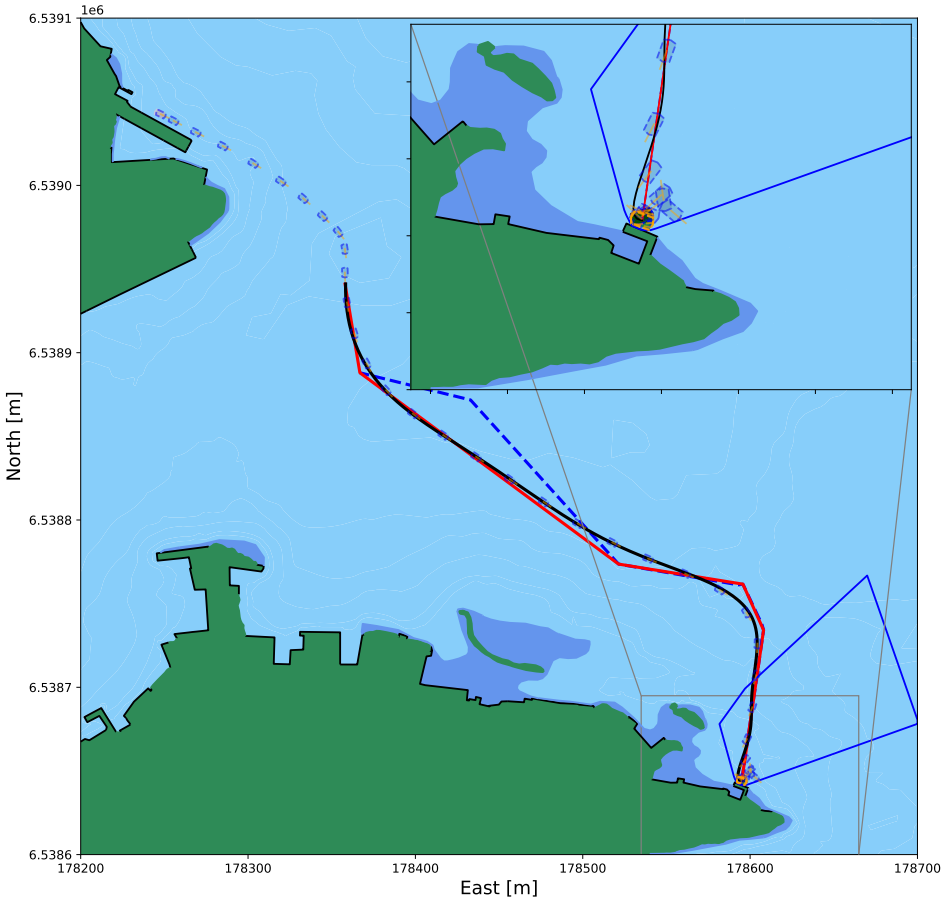


Figure 4.12: Complete trajectory of the give-way docking scenario, zoomed in on the final position. The final position is plotted in black, desired position in green, future predictions in orange, and shadow ships in gray.

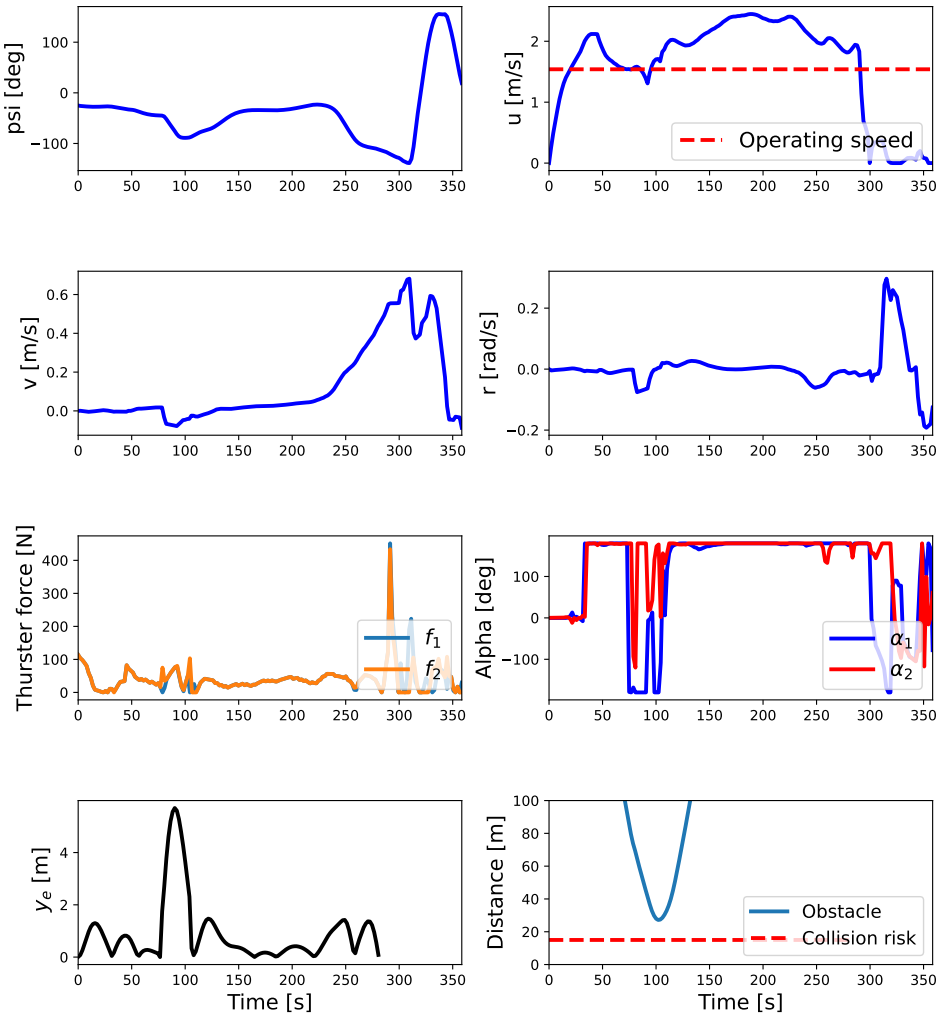


Figure 4.13: States η and ν, along with cross-track error and distance from obstacle when crossing and docking in a give-way scenario.

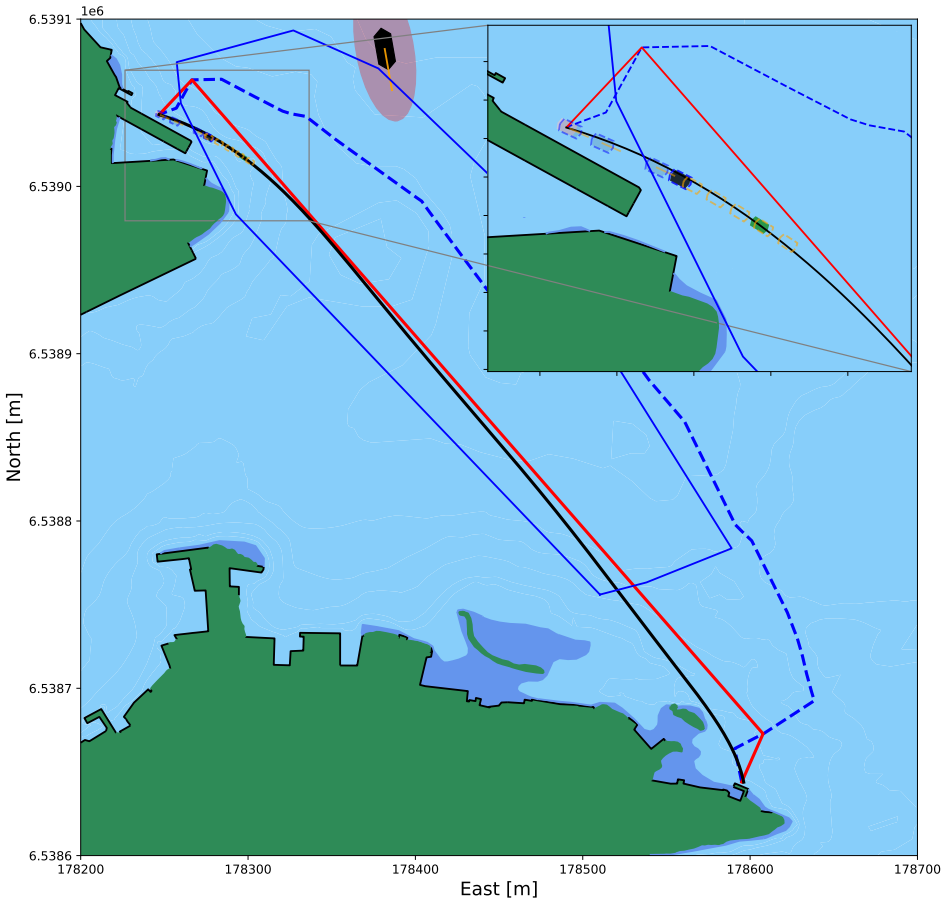


Figure 4.14: Trajectory of the overtaking scenario when the systems detect a collision risk. The current position is plotted in black, desired LOS-pose in green, future predictions in orange, and shadow ships in gray. The blue-lined convex set is the current safe operation area for the ASV

given more freedom to find the optimal control sequence. However, this approach could also lead to convergence to local minima. Consequently, it was imperative to monitor the success of the docking maneuver. The different states indicate the effort that was made around 295 seconds into the simulation in order to reach the global optima.

4.1.3 Overtake scenarios

Overtaking plays a vital role in marine navigation as it enables an ASV to pass another vessel while adhering to the COLREGs and maintaining efficient navigation. As discussed in Section 3.1.4, there are two possible overtaking scenarios for the OS. It can overtake the TS either on its port side or starboard side, depending on the TS's heading and its relative position to the OS' heading.

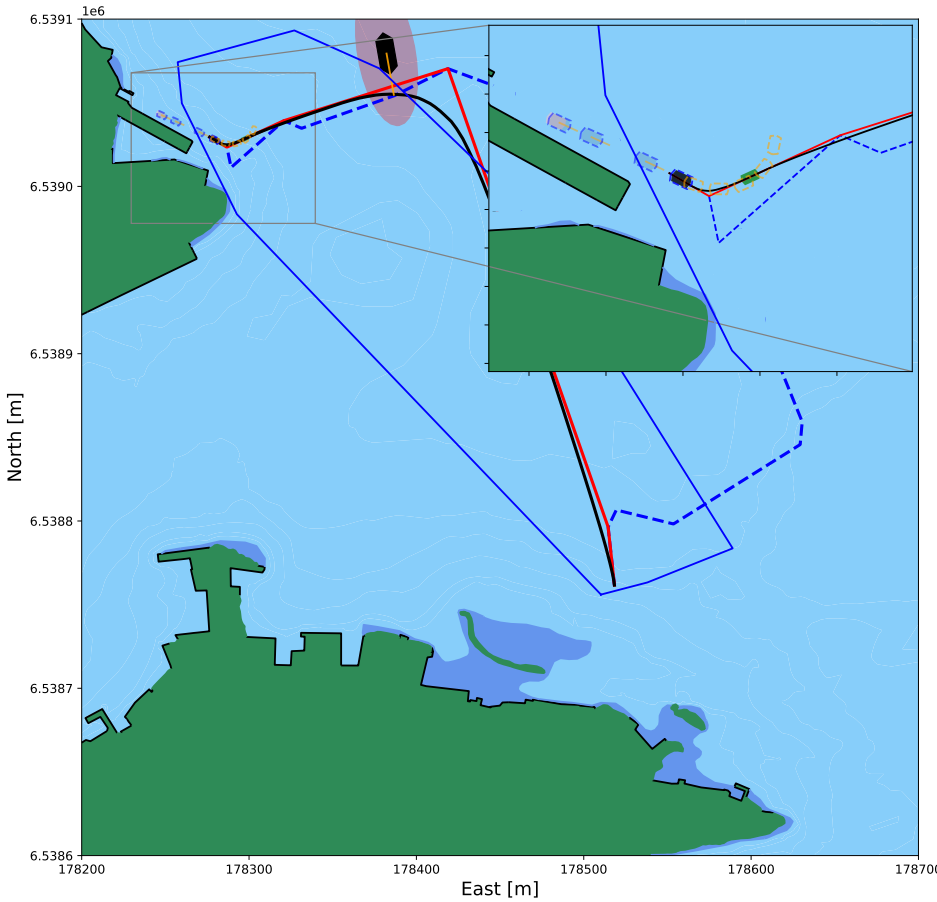


Figure 4.15: Illustration of the ASV's reaction to detecting a collision risk in a give-way scenario.

Overtaking on port side

Figure 4.14 illustrates the early detection of a collision risk by the ASV during the simulation. The replanned path was generated in 3.3 seconds and had a direction similar to that of the obstacle, as shown in Figure 4.15. The relatively long replanning time was due to the forbidden area ahead of the TS, which initially resulted in an optimal path going in the opposite direction of the desired position. According to the COLREGs, the overtaking vessel must continue to keep clear of the other vessel until it has passed completely. Additionally, since the TS and OS were traveling at approximately similar speeds, it was challenging for the OS to overtake. Therefore, the optimal path was not updated until around 275 seconds into the simulation, as depicted in Figure 4.16.

The states of the OS, along with the distance from the optimal path and TS, are presented in Figure 4.18. The ASV successfully docked at the desired position while keeping clear

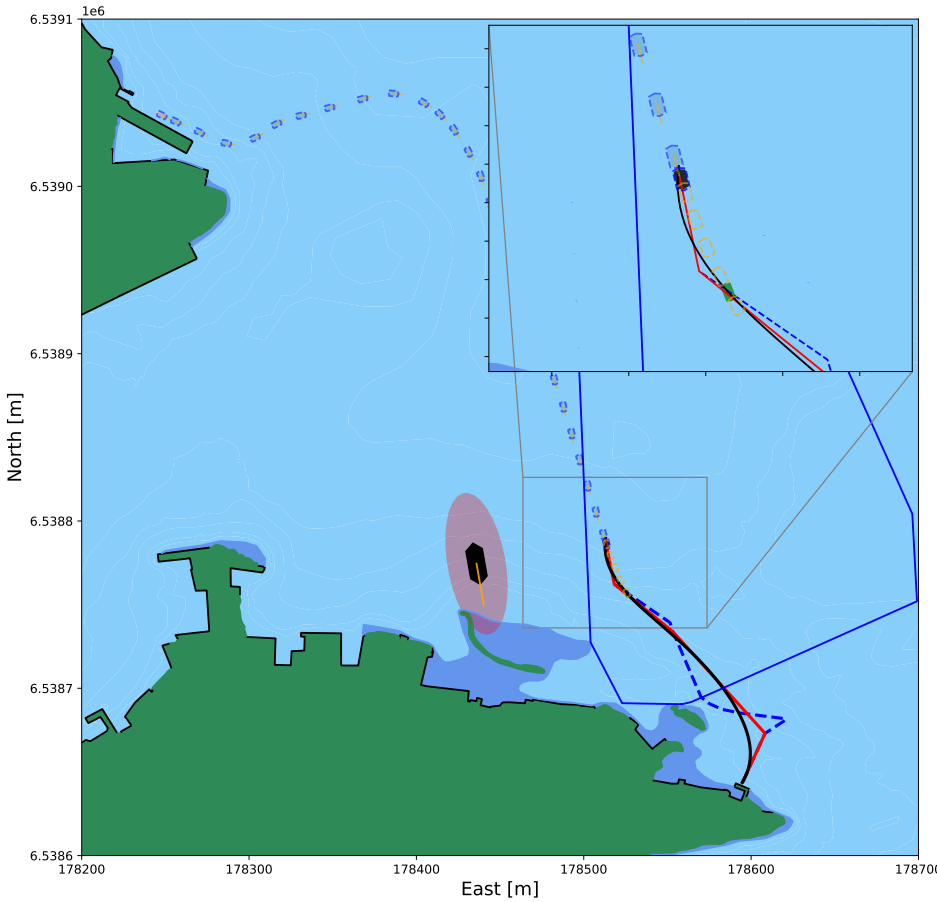


Figure 4.16: Illustration of the ASV updating the optimal path when the situation is considered safe during an overtaking scenario on the port side.

of the overtaken vessel, maintaining an average cross-track error of 0.65 meters. Throughout most of the simulation, the ASV maintained a speed close to the operational speed. The total simulation time was approximately 360 seconds, with a computation time of 160 seconds. This resulted in an average computational time of 0.44 seconds per iteration.

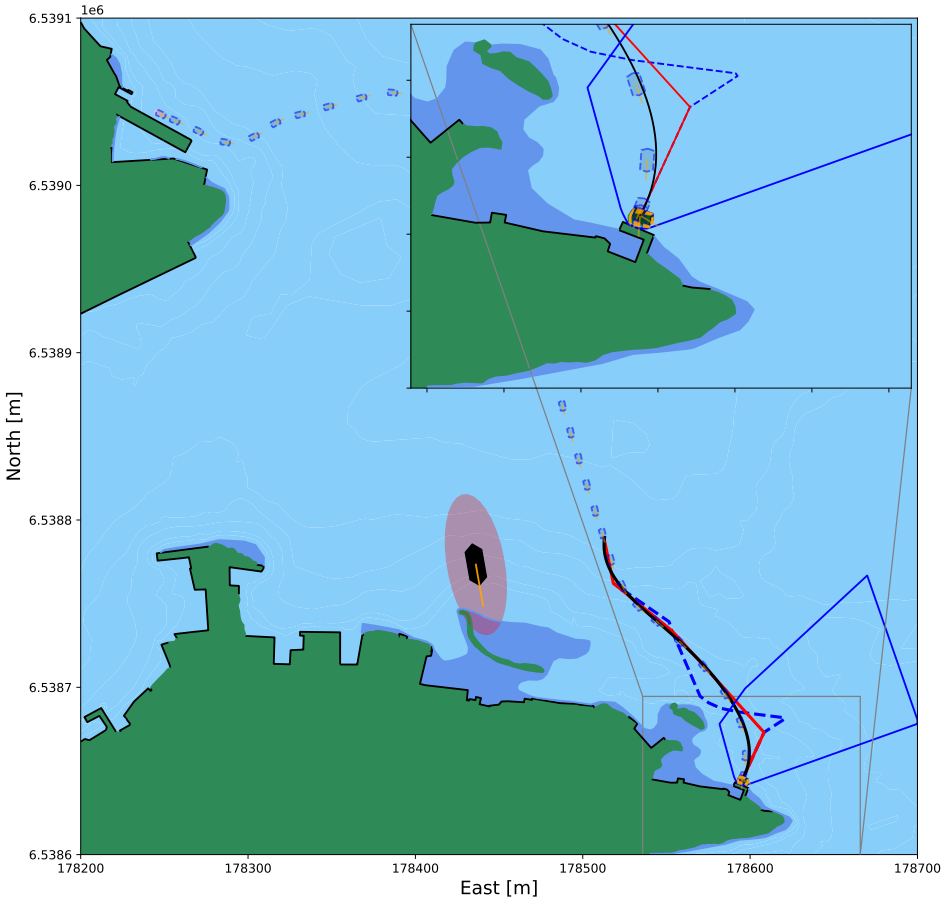


Figure 4.17: Complete trajectory of the overtaking scenario on the port side, zoomed in on the final position. The final position is plotted in black, desired position in green, future predictions in orange, and shadow ships in gray.

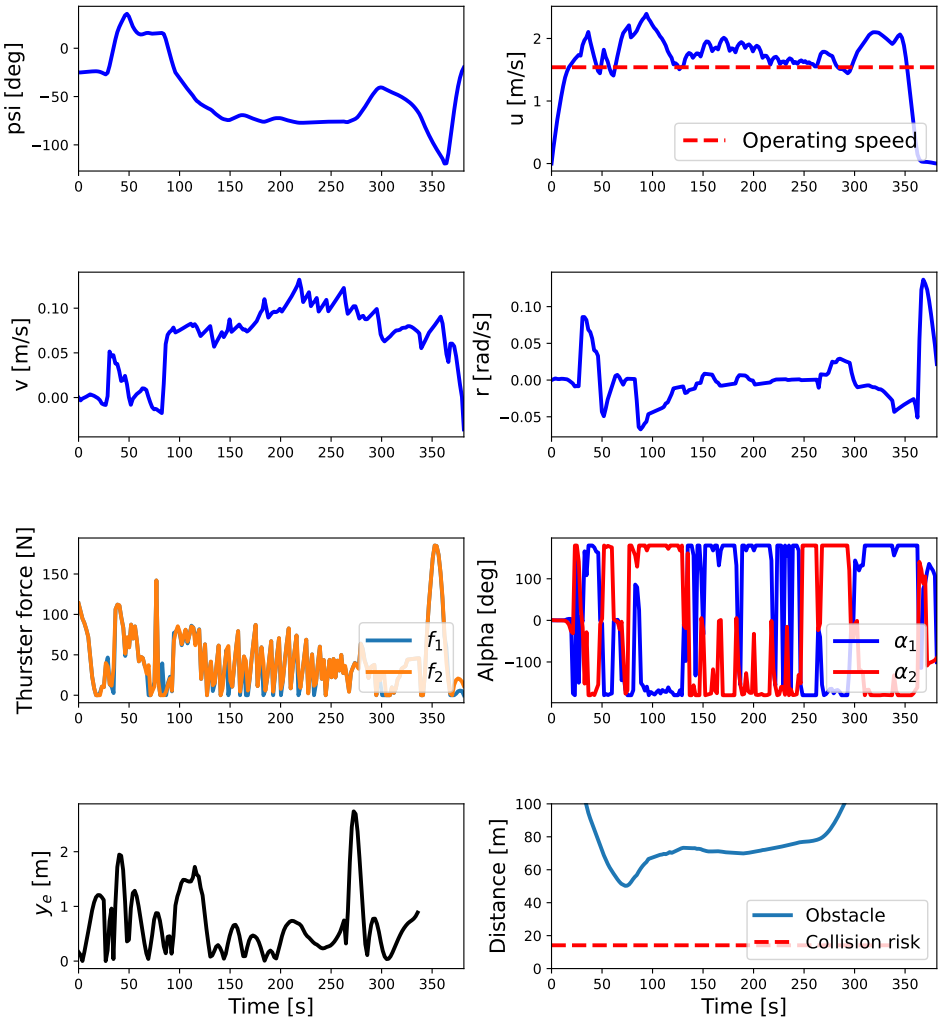


Figure 4.18: States η and ν, along with cross-track error and distance from the obstacle, during the crossing and docking in an overtaking scenario on the port side.

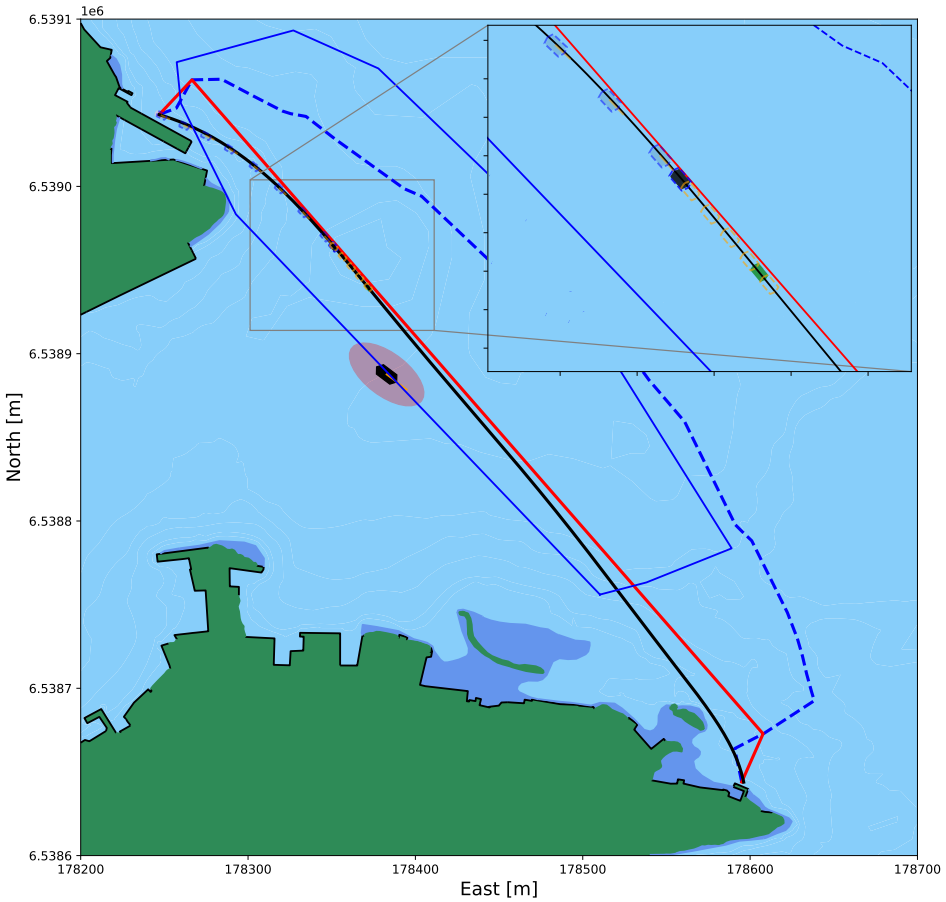


Figure 4.19: Trajectory of the overtaking scenario when the systems detect a collision risk. The current position is plotted in black, desired LOS-pose in green, future predictions in orange, and shadow ships in gray. The blue-lined convex set is the current safe operation area for the ASV.

Overtaking on starboard side

In the case where the OS is advised to overtake the TS on its starboard side, it shall keep out of the way of the vessel being overtaken. The TS in this scenario was traveling at a constant speed of 1.4 m/s, with dimensions of 15 meters in length and 7 meters in width. Figure 4.19 illustrates the time step before the ASV calculated the TCPA and DCPA to be below the risk threshold. The new optimal path was determined in less than 1 second. The path, as shown in Figure 4.20, was clearly directed towards the TS. However, it took into account the area that the TS would vacate as it moved forward. If the TS were to stop or reduce its speed, the collision risk would increase, requiring a new path calculation. Once the ASV passed the TS, it updated the optimal path in 1.4 seconds, as depicted in Figure 4.21. The ferry successfully docked at the jetty in 278 seconds. The simulation took 90

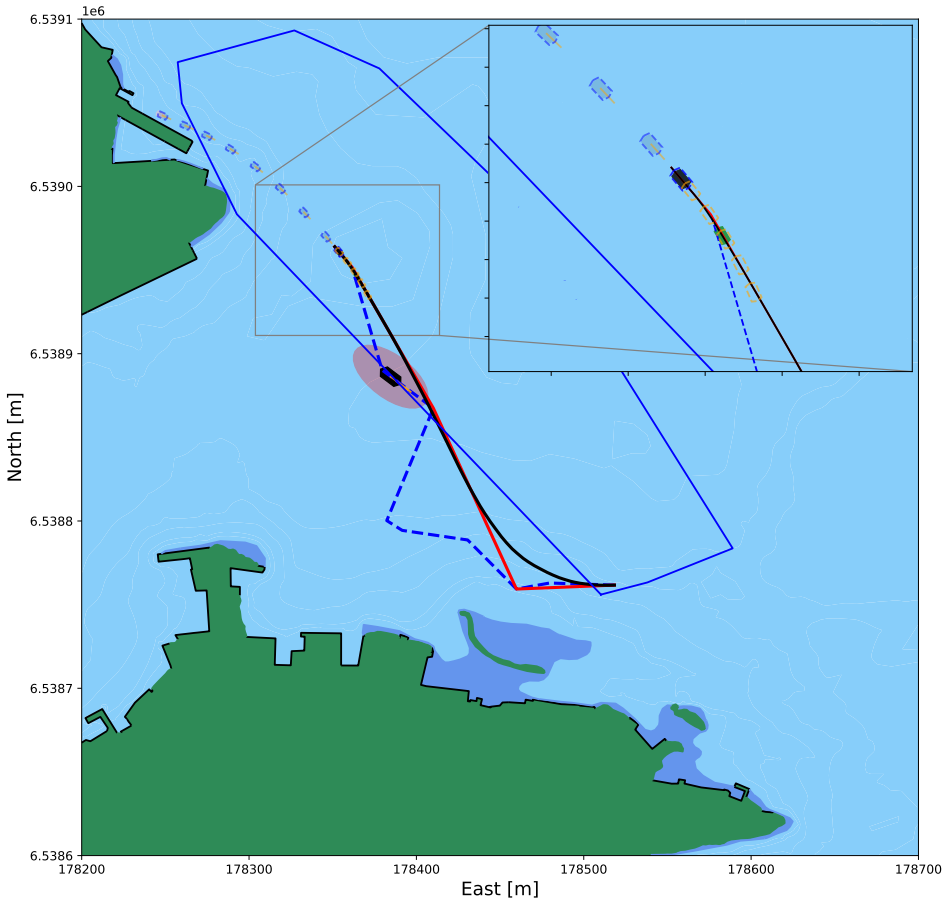


Figure 4.20: Illustration of the ASV's reaction to detecting a collision risk in an overtaking scenario.

seconds to compute, with 20 seconds spent in the docking phase. Resulting in an average computational time of 0.32 seconds per iteration in the NLMPC.

The optimal path through the simulation did not require any sharp turns, except closing in on the desired endpoint. In Figure 4.23, showing the ASV's states, it is clear that the vessel easily followed the optimal path, with only an average deviation of 0.24 meters. Moreover, the energy put into following the final turn is visible by examining the thruster force along with the increase in sway speed.

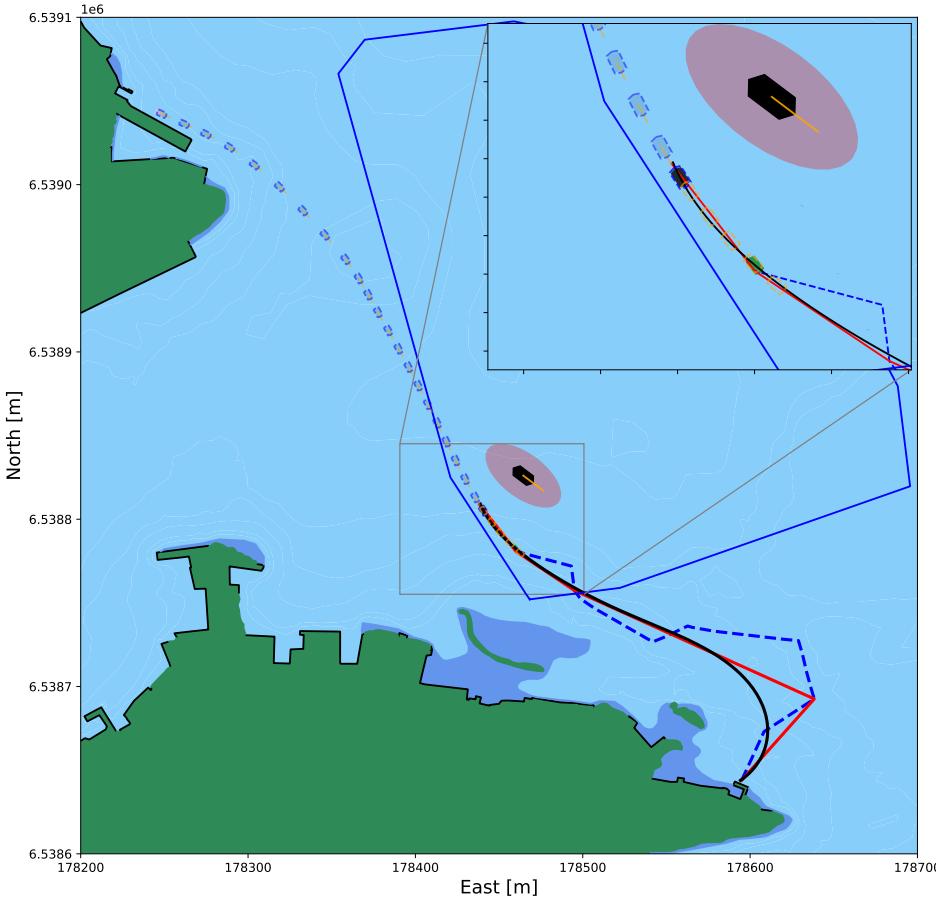


Figure 4.21: Illustration of the ASV updating the optimal path when the situation is considered safe.

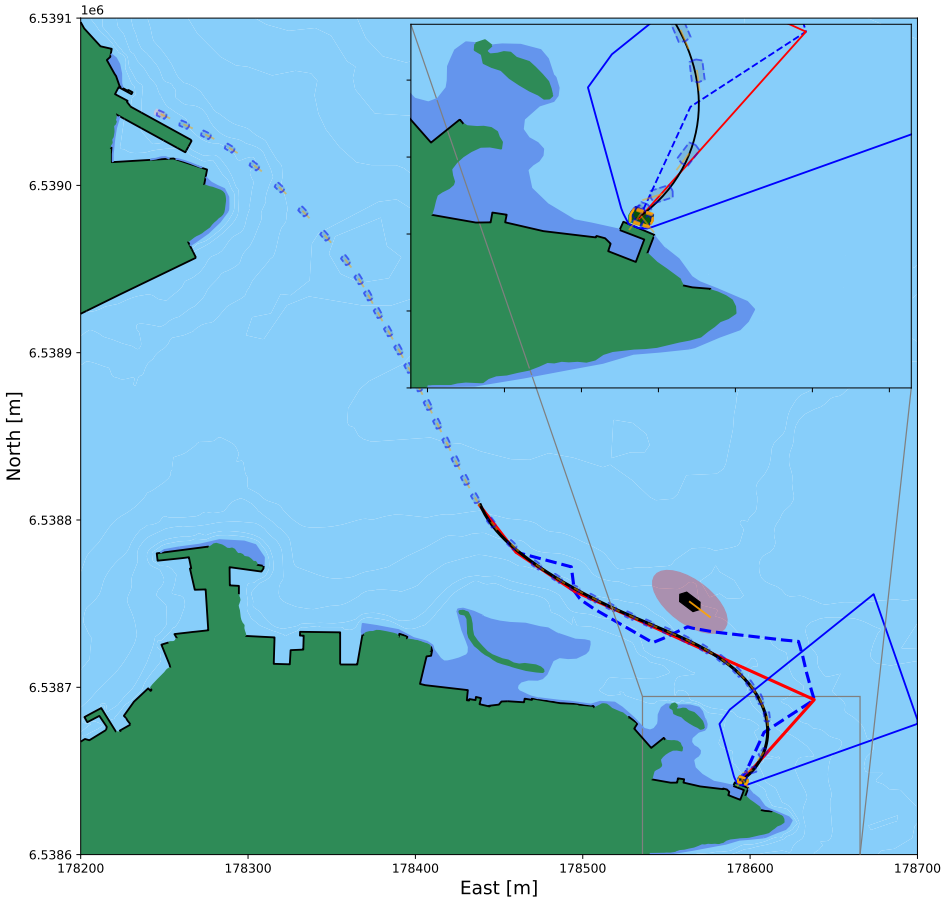


Figure 4.22: Complete trajectory of the give-way docking scenario, zoomed in on the final position. The final position is plotted in black, desired position in green, future predictions in orange, and shadow ships in gray.

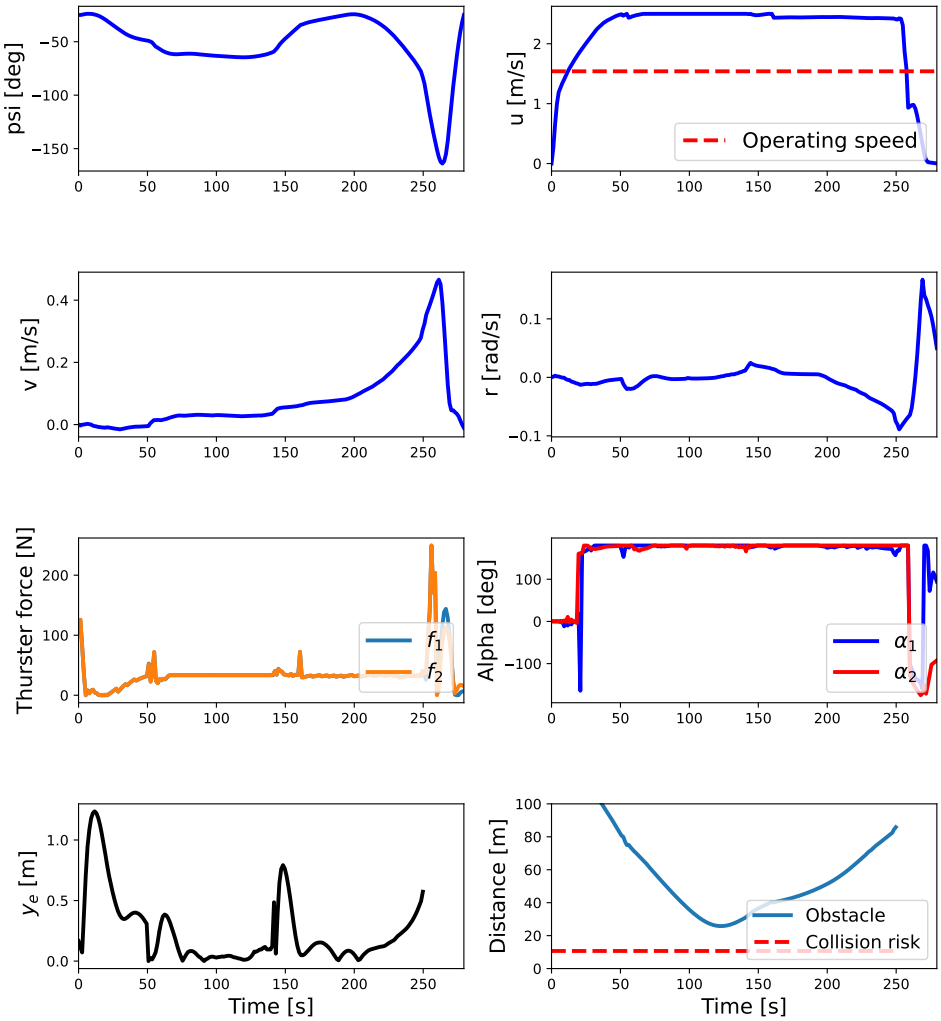


Figure 4.23: States η and ν, along with cross-track error and distance from obstacle when crossing and docking in a give-way scenario.

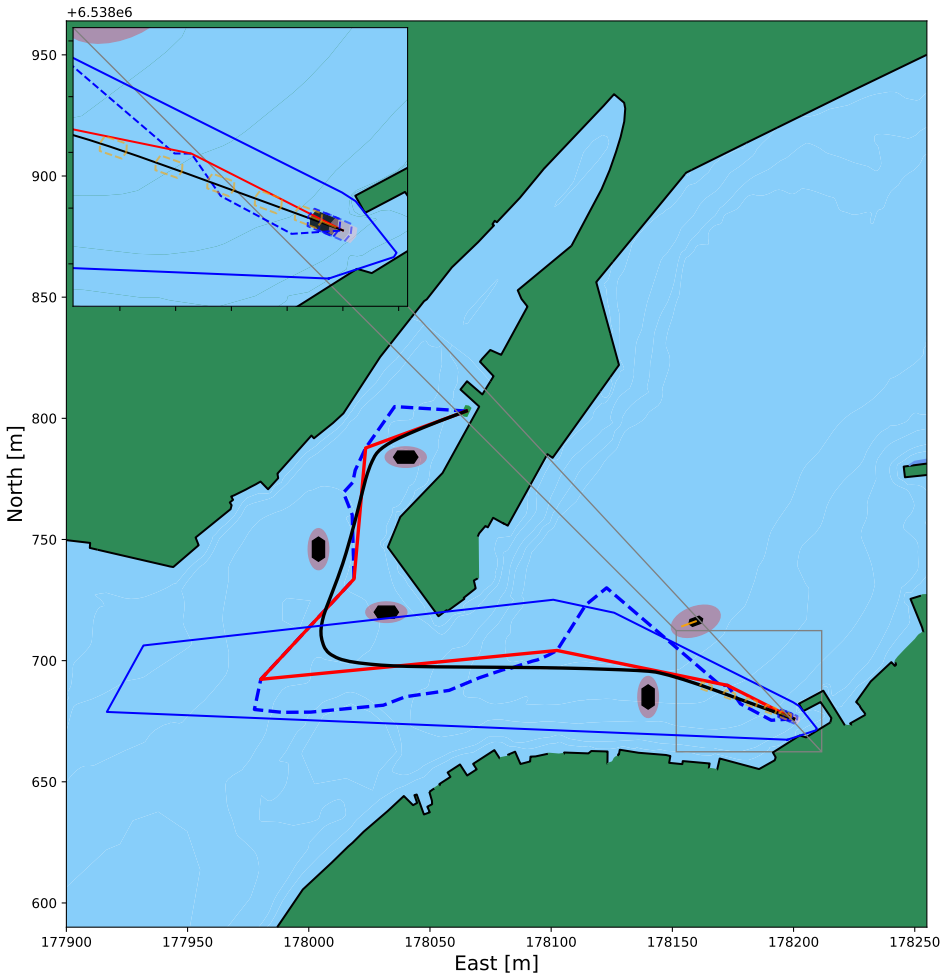


Figure 4.24: Illustration of the initial optimal path and first predicted states. Zoomed in on the current position (in black) and predicted states as orange dotted lines.

4.2 Complex scenario

In order to visualize the limitations of the proposed method, the ASV was simulated docking in a confined harbor area with several dynamic and static obstacles. In this scenario, the ASV avoided two dynamic obstacles, where it considered COLREG rules 13 and 17. Moreover, static obstacles were placed in proximity to the docking pose to increase difficulty.

In this scenario, the COLREG advised the ASV to overtake the first occurring TS, TS_1, on its starboard side. Figure 4.24 and 4.25 illustrate the scenario before and after the ASV

has replanned, respectively. The optimal path is updated, as shown in Figure 4.26 when the scenario is deemed safe. The replanning and updating of the optimal path took 2.2 and 4.4 seconds respectively. Indicating the computational complexity of finding the shortest path in a non-convex, complex area.

Moving forward, a dynamic obstacle, TS_2 crossing the harbor appeared ahead of the OS. The harbor and obstacles and their respective forbidden areas vacated the area towards the desired docking pose. The ASV was therefore set in a waiting state, as shown in Figure 4.27. While the vessel was in this state, it continuously updated the optimal path in order to move as close to the docking point as possible. These updating computations took between 1-2 seconds to compute, leading to a total computational time of 47 seconds while the vessel was in a waiting state. In Figure 4.28, the ASV updated the optimal path on its way behind the TS_s. Finally, in Figure 4.29 it is able to find a feasible path past the dynamic obstacles. This figure also illustrates a weakness in the smoothing algorithm. The reduced shortest path, depicted in red, has a small angle and the smoothing algorithm is limited by the movement constraint of the vessel. Resulting in a smooth path that has sharp angles. Utilizing a look-ahead distance large enough such that the vessel moves past such waypoints, is crucial for a smooth trajectory.

Figure 4.29 showcases that the ASV updated the optimal path again as it reached the end of the previous optimal path. It is also visible that the ASV is moving sideways as it moves close to the harbor and has limited maneuverability. Indicating that utilizing OCP also for the transit phase may render sub-optimal solutions. The final optimal trajectory with the respective states is illustrated in Figure 4.31 and 4.32, respectively. Figure 4.32 illustrates that the vessel is never at collision risk with the dynamic obstacles. On the other hand, it does travel in close proximity to especially the last obstacle. It is therefore vital to have a clear definition of static and dynamic obstacles in a practical sense. The spike in cross-track error occurs when the vessel is set in a waiting state and the optimal path is not directly connected to the ASVs current position. Rendering a faster convergence to this state.

The total simulation time took 283 seconds, while computing the entire scenario, took 392 seconds, including the 47 seconds spent on updating the optimal path in the waiting state. Such a long computational time is not ideal for real-world implementations. Measures must therefore be taken in order to reduce the computational time to render a feasible solution for such complex scenarios.

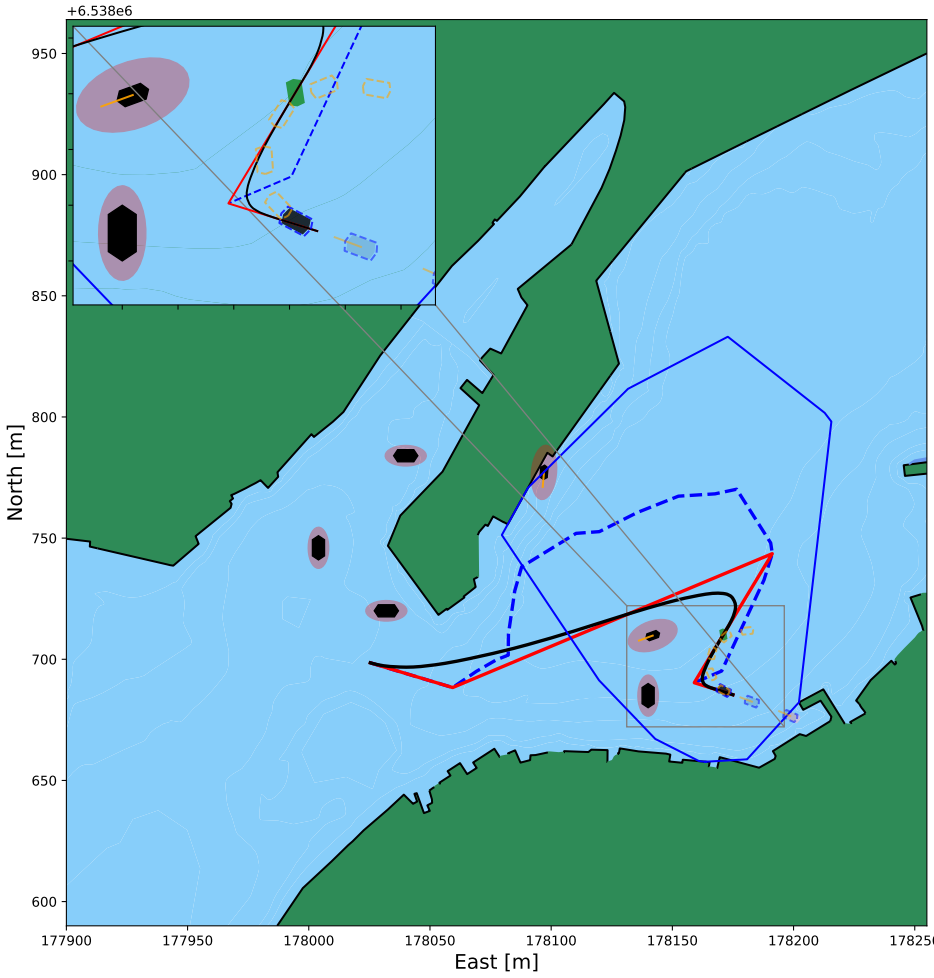


Figure 4.25: Replanning of the optimal path, as the ASV detects collision risk with the dynamic obstacle. It classifies the scenario as overtaking on the TS' starboard side (OT_s).

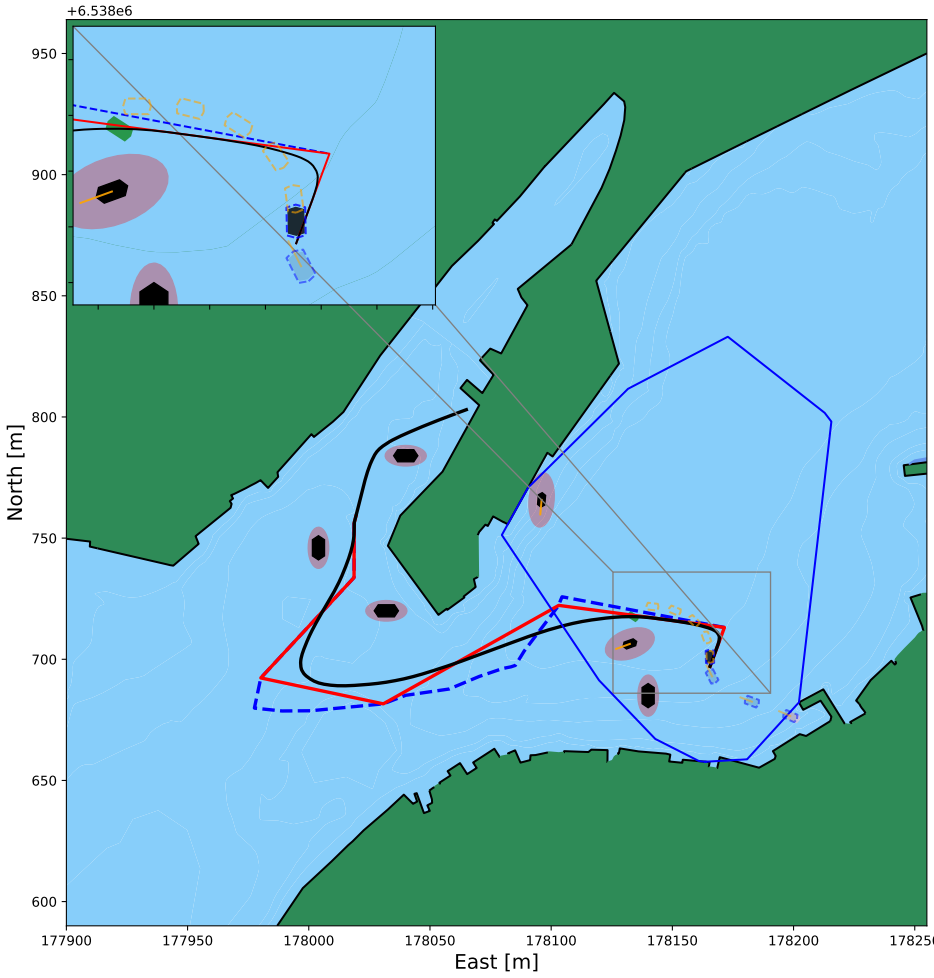


Figure 4.26: Illustration of the ASV updating the path as the relevant dynamic obstacle is no longer ahead of the OS. The OS still has to pass the TS on its starboard side.

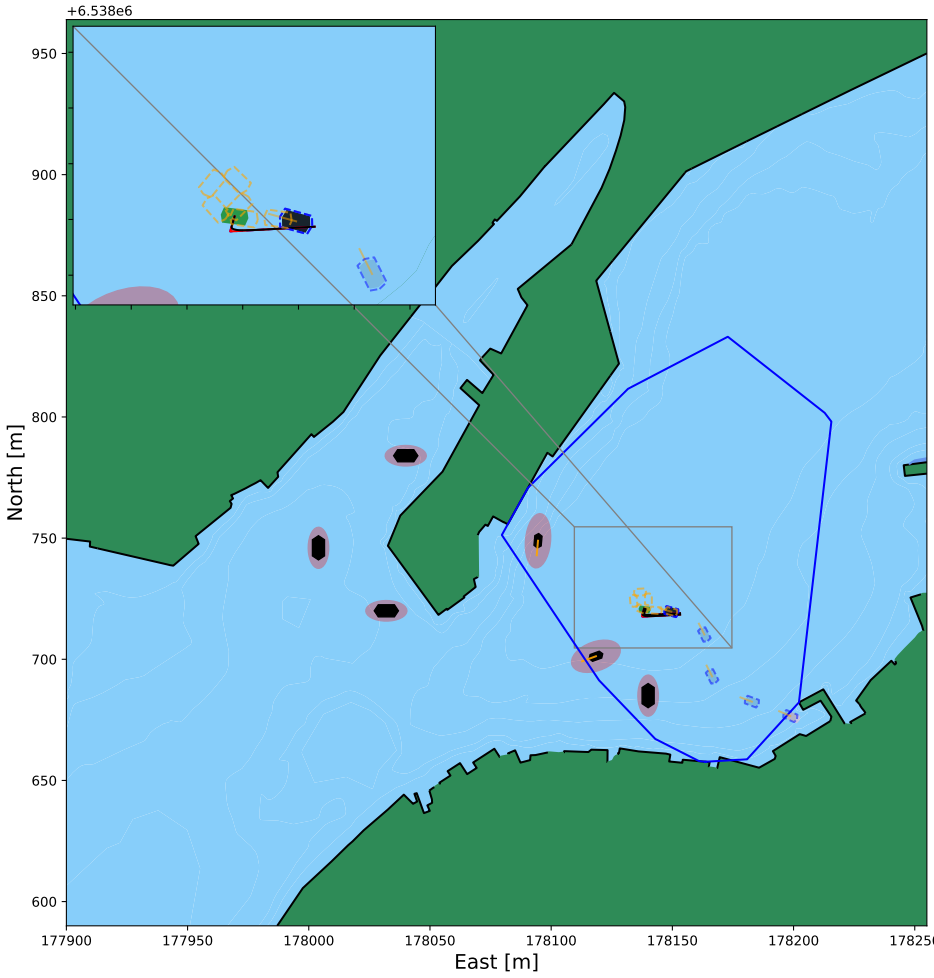


Figure 4.27: Illustration of the ASV detecting collision risk with another dynamic obstacle. The ASV classifies the scenario as a give-way scenario. With no feasible path, it is put in a wait state.

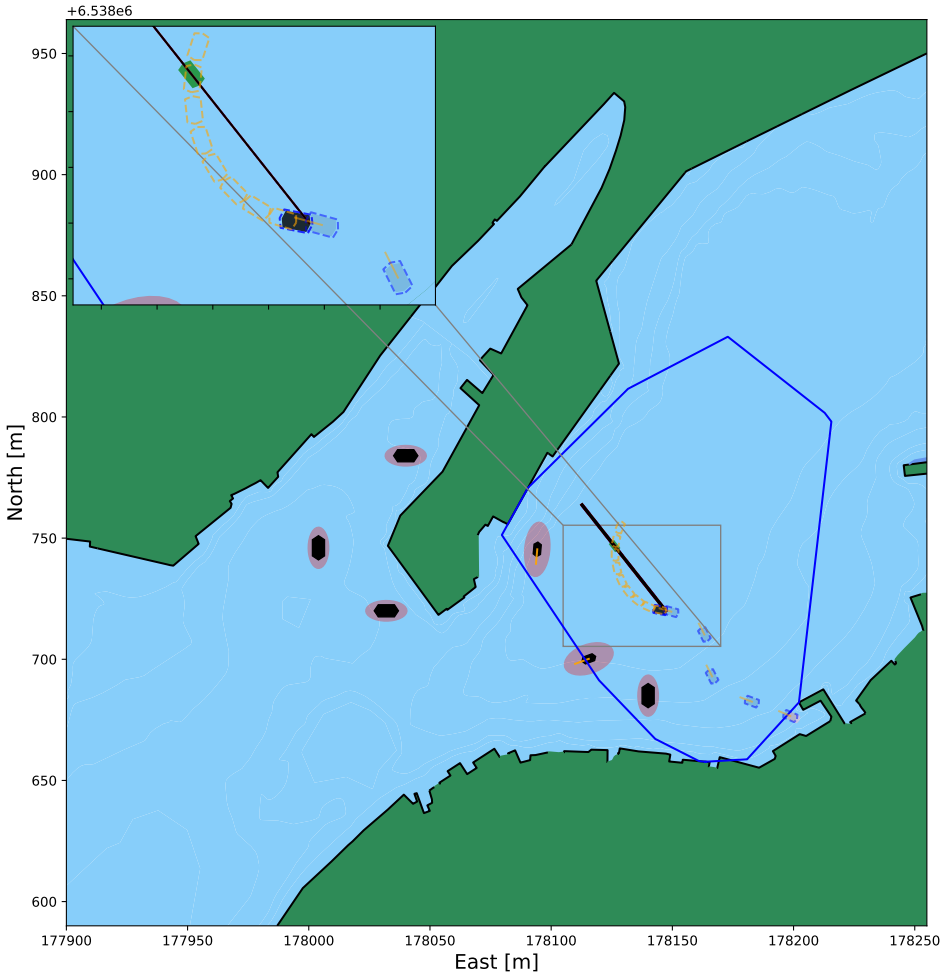


Figure 4.28: The ASV updates the optimal path, still giving way to the dynamic obstacle. However, moving closer to the endpoint.

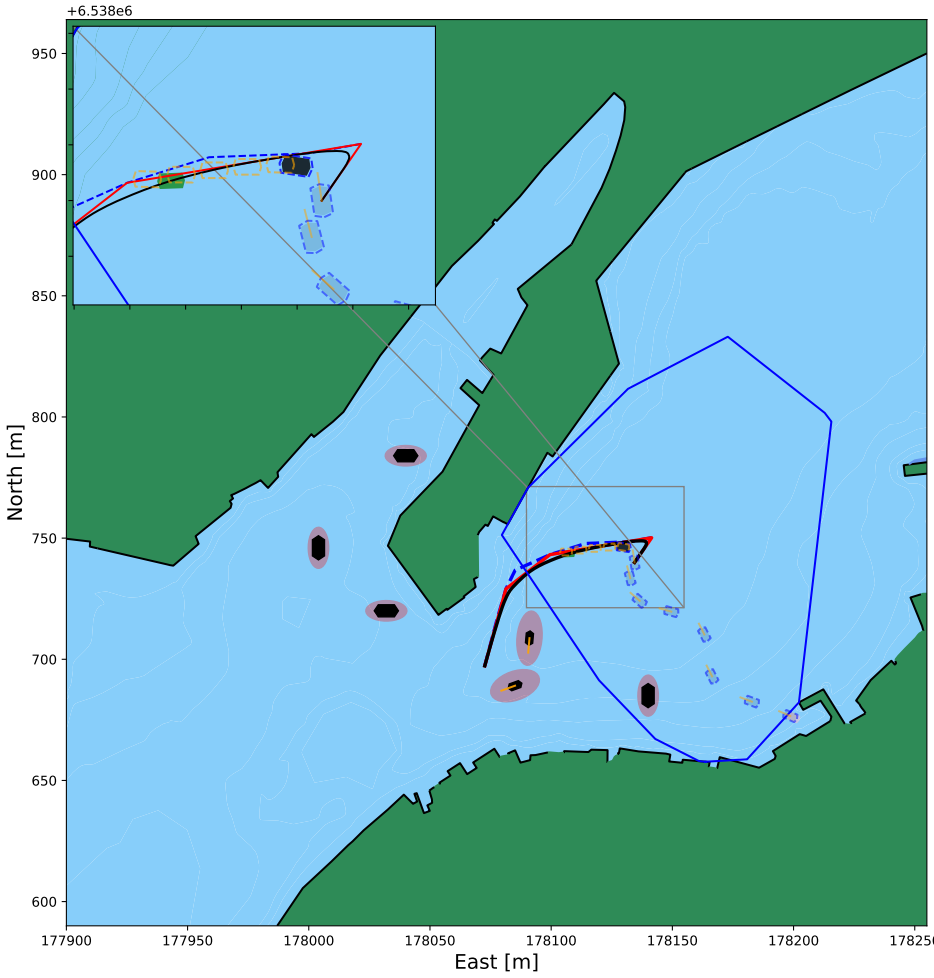


Figure 4.29: Visualization of the moment the ASV updates the optimal path and is able to find a feasible path past the dynamic obstacles.

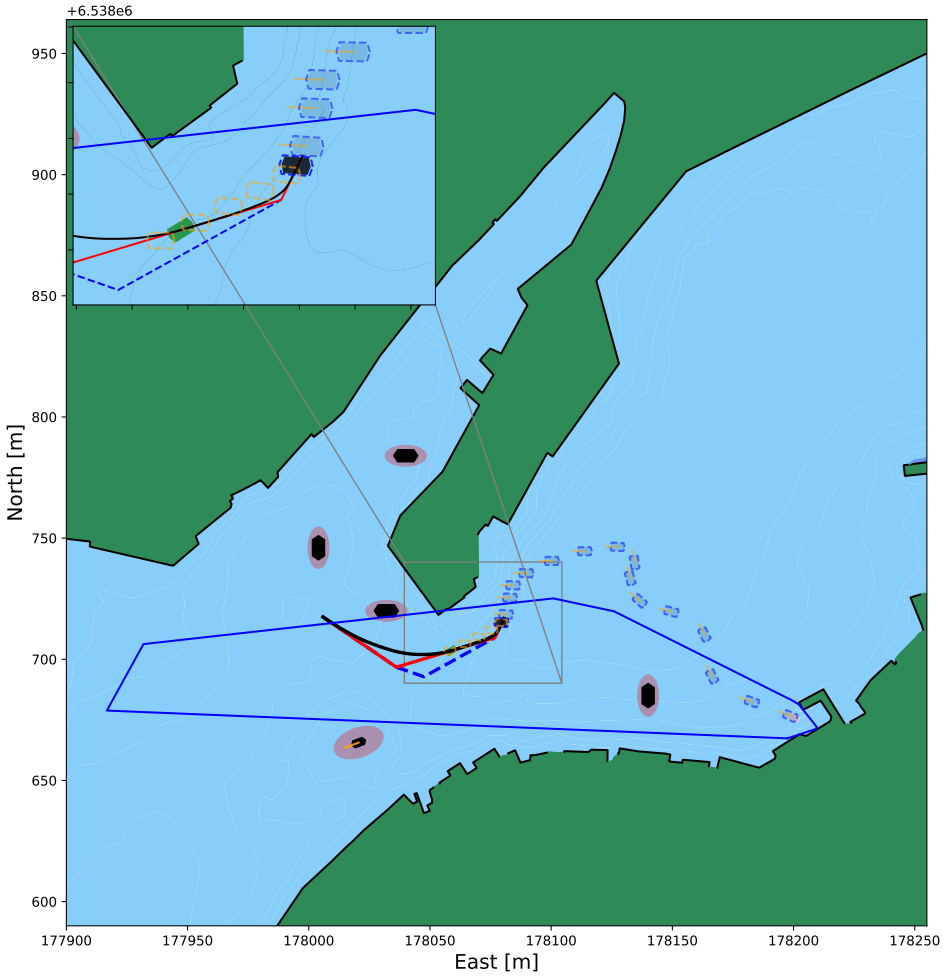


Figure 4.30: Figure depicting the ASV updating the optimal path as it has reached the end of the previous optimal path.

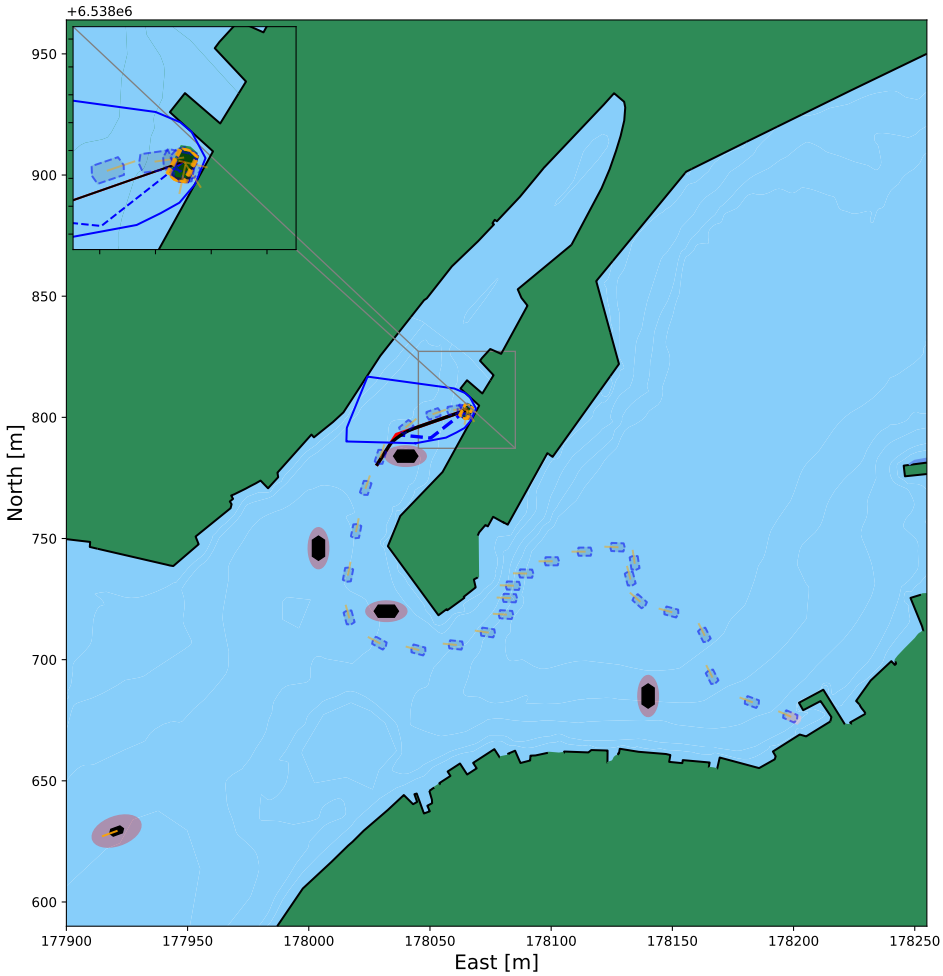


Figure 4.31: Complete trajectory of the give-way docking scenario, zoomed in on the final position. The final position is plotted in black, desired position in green, future predictions in orange, and shadow ships in gray.

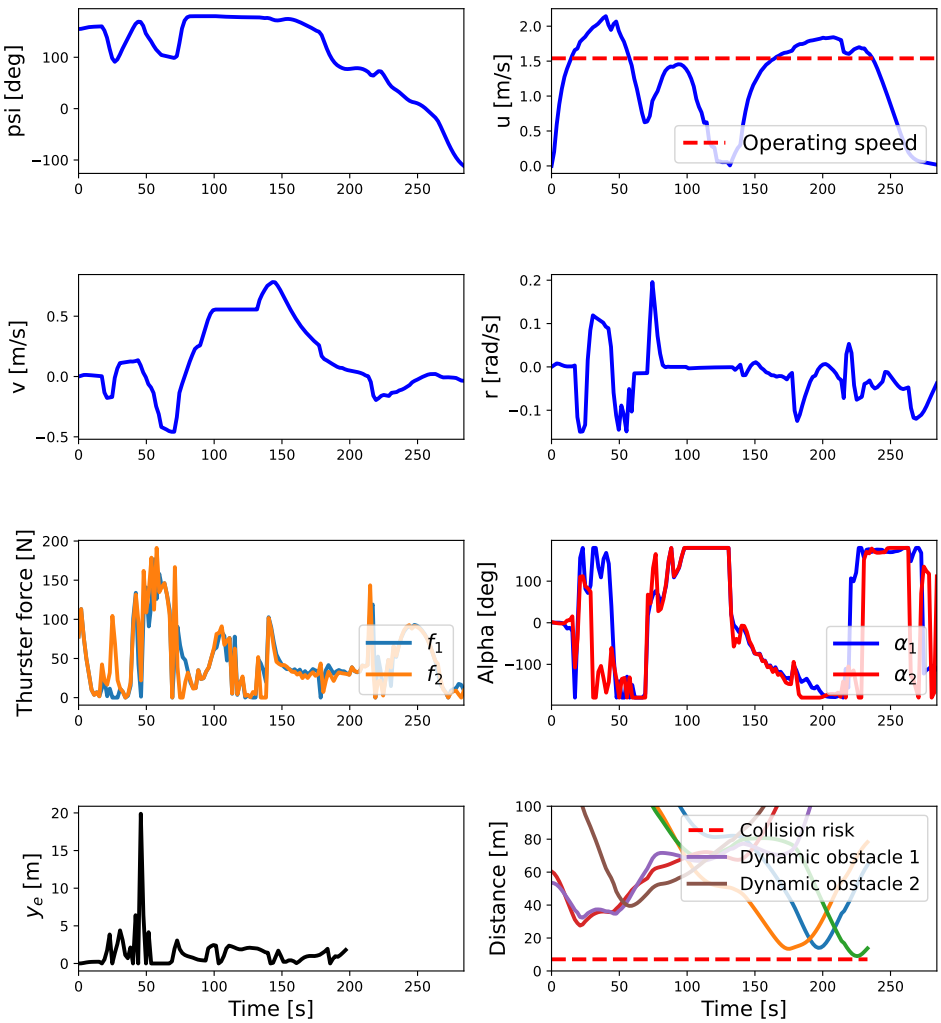


Figure 4.32: States η and ν, along with cross-track error and distance from obstacle when crossing and docking in a give-way scenario. The distance to the dynamic obstacles is shown in purple and brown. The remainder colors are distances to static obstacles.

4.3 Discussion

Autonomous docking systems can play a crucial role in ensuring safe and efficient vessel navigation while complying with the COLREGs. The simulations presented in this study demonstrate the system's ability to achieve COLREG compliance and perform safe docking maneuvers. The ASV effectively followed the relevant rules, maintained appropriate distances from other vessels, and successfully completed the docking procedures without compromising safety.

One of the core aspects of this research was to ensure safe docking while remaining compliant with COLREGs. However, implementing these regulations into the decision-making process of autonomous systems remains challenging. While it may be trivial to classify the applicable COLREG rules for a given situation, maneuvering in compliance with these rules remains a challenging task. The regulations are primarily designed for one-on-one encounters and often rely on qualitative descriptions, allowing humans to rely on experience and skills when assessing situations. In complex scenarios like crowded harbor areas, the interpretation of these regulations for an ASV can be ambiguous. Further research is needed to develop a framework facilitating safe and efficient maneuvering in trafficked areas. This would considerably enhance the safety of autonomous urban ferries in future applications.

One important aspect to consider in collision risk detection is the limitation of using the CPA as the sole measure. While CPA is commonly used, it relies on the assumption that the obstacle will maintain a constant heading and speed. This assumption does not always hold true, particularly in congested harbors where dynamic changes are more prevalent. Therefore, relying solely on CPA for collision risk detection may not provide reliable results. Future research should explore alternative methods that consider dynamic changes in the trajectories of obstacles to enhance collision risk detection accuracy. New research by Zhu and Ding (2023) proposes an algorithm for finding the optimal collision avoidance point, which may help assess the collision risk and take appropriate action.

In autonomous docking systems, ensuring computational feasibility is crucial. Path generation becomes more computationally demanding with an increasing number of waypoints, particularly evident in complex scenarios such as navigating through an archipelago. To address this issue, the replanned path was connected to the original optimal path at an appropriate point. Additionally, the replanning phase could run in parallel to the NLMPC. This approach could save time, provide a seamless transition between motion planning and control, and mirror a realistic operation for an ASV. Moreover, utilizing forbidden areas to represent different COLREG rules in order to find an optimal path, showed to be a promising approach to achieving an optimal path while considering the COLREG rules.

In terms of solving OCPs in a closed-loop MPC, balancing computational feasibility and accuracy is a significant challenge. Formulating the docking problem as an OCP offers an effective way of solving the control allocating problem and adhering to physical constraints while preserving the vessel's dynamics. On the other hand, the nature

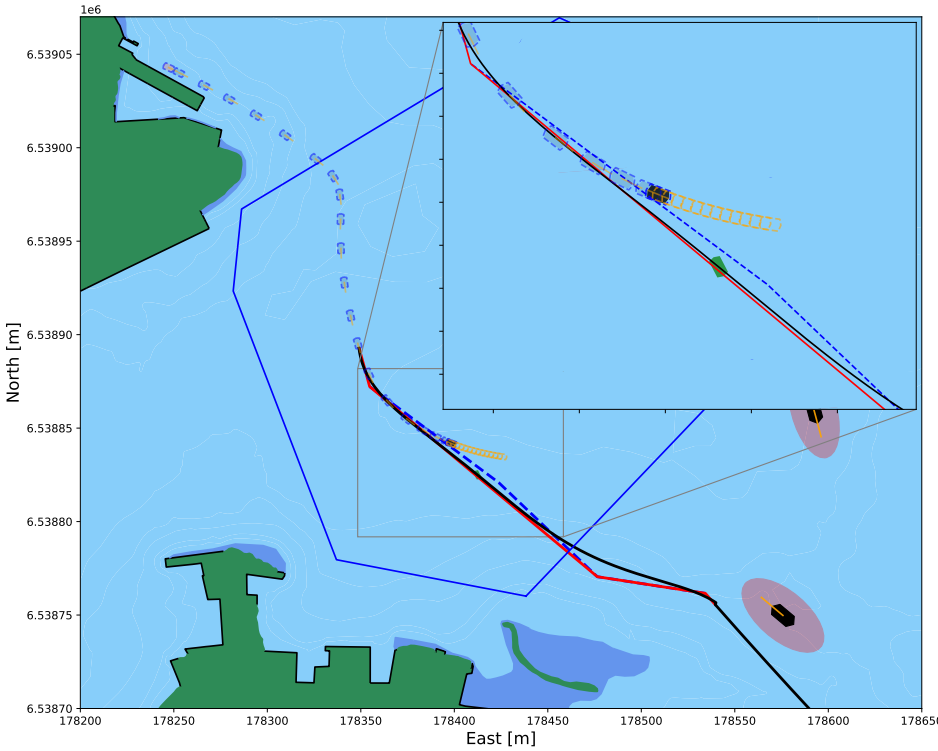


Figure 4.33: Illustration of the vessel diverging from the optimal path as the OCP converges to a local minimum.

of optimization-based control approaches introduces unpredictability and convergence to local minima. There are no guarantees for finding an optimal solution, particularly in scenarios where an optimal solution is imperative, such as in close proximity to obstacles. In an attempt to overcome this issue, the OCP was placed in a closed-loop MPC. However, by doing this, it becomes more dependent on finding feasible trajectories. Increasing the computational accuracy drastically increases the computational complexity. This computational complexity can pose challenges in real-time decision-making.

Tuning the control parameters can be a complex task, as it involves considering various factors such as lookahead distance in LOS guidance, time horizon and step size, and the different cost functions. Figure 4.33 showcases a scenario where the vessel is not able to follow the desired path as it continuously converges to a local minimum. When the fully-actuated vessel is in transit, it may be given too much room to find an optimal solution and may end up converging to a local optimum. It could therefore be beneficial to utilize an underactuated model, such as in Ødven et al. (2022), to render better solutions while in transit. Moreover, deviation from the desired path could also be tackled by implementing terminal cost in the OCP. By incorporating a terminal cost, the OCP can prioritize achiev-

ing a particular final state while considering other objectives and constraints. It provides a mechanism to guide the system's behavior toward a desired outcome [77, 78, 79].

In practical scenarios, a combination of longer prediction horizons and a Proportional–Integral–Derivative (PID) controller has proven effective in handling path deviations and ensuring accurate trajectory following. Furthermore, parallel optimization approaches, such as finding optimal paths in separate threads, can help improve control performance and minimize deviations. Alternatively, methods such as RL, PID, and sliding mode control be utilized to follow the optimal path, while the OCP could generate an optimal control sequence in order to successfully dock the vessel [10, 80].

5

Conclusion and future work

This thesis successfully integrates a COLREGs-aware collision avoidance (COLAV) system, which combines computational geometry and nonlinear optimal control, to facilitate safe docking and obstacle avoidance. The simulations conducted provided evidence of successful docking and obstacle avoidance across various scenarios, demonstrating the viability of formulating the docking problem as an Optimal Control Problem (OCP). Additionally, the simulations illustrated the vessel effectively utilizes the azimuth features, executing a docking maneuver resembling human-like behavior.

In terms of methodology, this study adopts techniques from Bitar et al., Martinsen, Thyri (2019, 2021, 2022), and Ødven et al. (2022) to tackle the docking problem. The process commences with the triangulation of the non-convex area to generate feasible waypoints that circumvent obstacles. Subsequently, an optimal, smooth path is determined through an A* search algorithm, a waypoint-reduction algorithm, and a path-smoothing algorithm. Convex sets are auto-generated, yielding a feasible area for the vessel's maneuvers. Lastly, an OCP is implemented within a closed-loop Model Predictive Control (MPC) to track the optimal path and perform human-like docking maneuvers. The computational complexity arises from the nonlinear vessel model, and representing obstacles and harbor area as spatial constraints in the docking scenario.

While the method proposed in this thesis has shown proficiency in docking across numerous scenarios, it is not without limitations. Nonlinear optimal control problems have a natural tendency to converge to a local optimum. Even though initializing the OCP with an optimal path is an attempt to mitigate this issue, it does not guarantee that the OCP will find the global optimum. Consequently, the formulation of efficient cost functions and the inclusion of terminal cost could be an important aspects to consider in future implementations.

Furthermore, the method's reliance on the Closest Point of Approach (CPA) for collision risk assessment, coupled with its assumption of constant speed and heading for dynamic

obstacles, oversimplifies the complexities of maneuvering in a congested harbor area. Despite providing a robust testing ground for COLREG-aware methods, a more comprehensive collision assessment logic is crucial for real-world Autonomous Surface Vessel (ASV) implementation

Additionally, the method does not account for external disturbances. Research by Martinsen et al. (2022) explored RL-MPC for tracking control of ASVs, and it would be intriguing to test the incorporation of Reinforcement Learning based Model Predictive Control (RL-MPC) tracking control instead of MPC for docking scenarios in future work. This could yield a solution that is more resilient to uncertainties and external disturbances, providing a practical solution that is not reliant on calm winds and currents.

In conclusion, the proposed method shows promising results as a way of solving the docking scenario with obstacle avoidance. It is able to efficiently maneuver to the docking point, and successfully dock while avoiding any obstacles by accounting for international regulations. Furthermore, this thesis serves as a steppingstone in the ongoing pursuit of developing viable and efficient methods for safely docking an ASV while considering its surroundings and maritime obligations, with several interesting and feasible approaches for further enhancing the method in the future, potentially yielding practical and promising results.

5.1 Future work

The proposed method presents various intriguing aspects that have the potential for further improvement to optimize the docking scenario. The following are recommendations for enhancing these aspects:

- Implement a PID controller in parallel with an MPC to ensure a more robust path-following sequence. This would facilitate the vessel's response to real-time changes, ultimately enhancing maneuverability and docking precision.
- Introduce a terminal cost in the OCP to generate a trajectory more likely to achieve the global optimum. By doing so, the system would have a higher chance of converging to the best solution, improving the vessel's docking efficiency and accuracy.
- Investigate more advanced collision risk assessment techniques that do not rely exclusively on the CPA. Implementing more sophisticated methods would account for dynamic changes in the harbor environment, thus offering a more robust and reliable collision prediction system.
- Investigate an RL-MPC tracking controller instead of PID controller. This approach could help create a docking system for an ASV that is more resilient to model uncertainties and external disturbances. By improving the system's ability to adapt to unexpected changes, the ASV would become more reliable and safe in real-world, unpredictable conditions.

- Investigate the application of Machine Learning (ML) methods to find an optimal path that complies with COLREG rules. The use of ML could provide a more nuanced understanding of the various constraints and considerations that come into play when determining the optimal path, leading to more effective navigation and improved adherence to maritime regulations.
- Conduct real-world testing of the proposed system to assess how it performs under various conditions.

Bibliography

[1] 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*, volume 55. IFAC PapersOnline, 2022. URL https://www.proceedings.com/content/067/067080webtoc.pdf.

[2] Andreas B. Martinsen, Anastasios M. Lekkas, and Sébastien Gros. *Autonomous docking using direct optimal control*. IFAC-PapersOnLine, 2019.

[3] Glenn I. Bitar, Vegard N. Vestad, Anastasios M. Lekkas, and Morten Breivik. *Warm-Started Optimized Trajectory Planning for ASVs*, volume 52. IFAC-PapersOnLine, 2019. URL https://www.sciencedirect.com/science/article/pii/S2405896319322104.

[4] Henrik R. Nordhus. *Automatic docking and obstacle avoidance for autonomous surface vessels*. (Project report in TTK4550) NTNU, 2022.

[5] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. *CasADi – A software framework for nonlinear optimization and optimal control*. Mathematical Programming Computation, 2018.

[6] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, West Sussex, United Kingdom, 2011.

[7] Idun Haugan. *NTNU trials world's first urban autonomous passenger ferry*. Norwegian SchiTech News, 2022. URL https://norwegianscitechnews.com/2022/09/ntnu-trials-worlds-first-urban-autonomous-passenger-ferry/.

[8] Petter K. Ødven. *(Master's thesis) Static and dynamic multi-obstacle avoidance and docking of ASVs using computational geometry and numerical optimal control.* NTNU, 2022.

[9] Anders A. Pedersen. *(Master's thesis) Optimization Based System Identification for the milliAmpere Ferry*. NTNU, 2019.

[10] Anastasios M. Lekkas and Thor Inge Fossen. A time-varying lookahead distance guidance law for path following. *IFAC Proceedings Volumes*, 45: 398–403, 2012. URL https://www.semanticscholar.org/paper/A-Time-Varying-Lookahead-Distance-Guidance-Law-for-Lekkas-Fossen/92416a61a4208011979b8155b229d1e2e8af4f73.

[11] Bjørn-Olav H. Eriksen, Glenn I. Bitar, Morten Breivik, and Anastasios M. Lekkas. *Hybrid Collision Avoidance for ASVs Compliant with COLREGs Rules 8 and 13-17*. Frontiers in Robotics and AI, 2020. URL http://hdl.handle.net/11250/2641164.

[12] Arnfinn Oksavik, Hans P. Hildre, Yushan Pan, Ian Jenkinson, Barbara Kelly, Dimitrios Paraskevadakis, and Robyn Pyne. *Future skill and competence needs*. NTNU, Available from: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2648963 (Downloaded: 21.09.2022), 2020.

[13] Allianz. *Safety and Shipping Review: An annual review of trends and developments in shipping losses, risk challenges and safety*. Available from: https://www.agcs.allianz.com/news-and-insights/reports/shipping-safety.html, 2022.

[14] The Research Council of Norway. *SFI Centre for Research-based Innovation*. 2019. URL https://www.forskningsradet.no/en/call-for-proposals/2019/centre-for-research-based-innovation/. (Visited: 13.12.2022).

[15] Det Kongelige Nærings- og Fiskeridepartement. volume 10. Stortingsmelding, 2020. URL https://www.regjeringen.no/contentassets/391f633b512b4866a4193ba67be27c3b/no/pdfs/stm202020210010000dddpdfs.pdf. (Visited: 18.12.2022).

[16] Finferries. *Finferries' Falco world's first fully autonomous ferry*. 2018. (Visited: 13.12.2022).

[17] Kongsberg. *Autonomous shipping*. 2020. (Visited: 07.12.2022).

[18] Naida H. Prevljak. *Yara Birkeland, world's 1st zero-emission containership, completes maiden voyage*. Offshore Energy, 2021. (Visited: 13.12.2022).

[19] Yara. *Crown Prince and youths christen world's first emission-free container ship*. 2022. (Visited: 13.12.2022).

[20] Kongsberg. *Autonomous ship project, key facts about Yara Birkeland*. 2022. URL https://www.kongsberg.com/maritime/support/themes/autonomous-ship-project-key-facts-about-yara-birkeland/. (Visited 21.01.2023).

[21] Katrine Damkjær. Technical University of Denmark, 2022. URL https://www.dtu.dk/english/news/all-news/dks-foerste-foererloese-faerge?id=552dbfa6-4b82-487c-9147-8c5ee85ffeac. (Visited: 18.12.2022).

[22] Orca AI. *Enhancing Safety for Maran Tankers Management - A Global Crude Shipping Powerhouse*. Available from: https://global-uploads.webflow.com/5e9875d488a340acfb8d2a88/63201785a7b0ab3420fc26f7_Maran%20Tankers%20Management%20Case%20Study%20light.pdf (Downloaded: 28.09.2022), 2022.

[23] Scooter Doll. *Autonomous cargo ship completes 500 mile voyage, avoiding hundreds of collisions*. 2022. (Visited: 28.09.2022).

[24] BBC. *AI-driven robot boat Mayflower crosses Atlantic Ocean*. 2022. URL https://www.bbc.com/news/uk-england-devon-61710706.

[25] Glenn I. Bitar, Andreas B. Martinsen, Anastasios M. Lekkas, and Morten Breivik. *Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments*, volume 53. IFAC-PapersOnLine, 2020. doi: https://doi.org/10.1016/j.ifacol.2020.12.1451. URL https://www.sciencedirect.com/science/article/pii/S2405896320318632. 21st IFAC World Congress.

[26] G.J.S. Rae and S.M. Smith. *A Fuzzy Rule Based Docking Procedure For Autonomous Underwater Vehicles*. 1992.

[27] H. Yamato. *Automatic berthing by the neural controller*, volume 3. Proc. of Ninth Ship Control Systems Symposium, 1990.

[28] Ella-Lovise H. Rørvik. *Automatic Docking of an Autonomous Surface Vessel*. (Master's thesis) NTNU, Trondheim Norway, 2020.

[29] Andreas B. Martinsen, Anastasios M. Lekkas, and Sébastien Gros. *Reinforcement learning-based NMPC for tracking control of ASVs: Theory and experiments*, volume 120. Control Engineering Practice, 2022. URL https://www.sciencedirect.com/science/article/pii/S0967066121002823.

[30] Kouki Wakita, Youhei Akimoto, Dimas M. Rachman, Yoshiki Miyauchi, Umeda Naoya, and Atsuo Maki. Collision probability reduction method for tracking control in automatic docking / berthing using reinforcement learning, 2022. URL https://arxiv.org/abs/2212.06415.

[31] Andreas B. Martinsen, Glenn I. Bitar, Anastasios M. Lekkas, and Sébastien Gros. *Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments*, volume 8. 2020.

[32] Yoshiki Miyauchi, Ryohei Sawada, Youhei Akimoto, Naoya Umeda, and Atsuo Maki. *Optimization on planning of trajectory and control of autonomous berthing and unberthing for the realistic port geometry*, volume 245. Ocean Engineering, 2022. doi: https://doi.org/10.1016/j.oceaneng.2021.110390. URL https://www.sciencedirect.com/science/article/pii/S0029801821016826.

[33] Dimas M. Rachman, Atsuo Maki, Yoshiki Miyauchi, and Naoya Umeda. *Warm-started semionline trajectory planner for ship's automatic docking (berthing)*, volume 252. Ocean Engineering, 2022. doi: https://doi.org/10.1016/j.oceaneng.2022.

111127. URL https://www.sciencedirect.com/science/article/pii/S0029801822005388.

[34] Shengke Ni, Ning Wang, Ziyi Qin, Xihan Yang, Zhengjiang Liu, and Haijiang Li. *A distributed coordinated path planning algorithm for maritime autonomous surface ship*, volume 271. Ocean Engineering, 2023. URL https://doi.org/10.1016/j.oceaneng.2023.113759.

[35] Bjørn-Olav H. Eriksen. *(PhD thesis) Collision avoidance and motion control for autonomous surface vehicles*. NTNU, 2019. URL http://hdl.handle.net/11250/2616394.

[36] Emil H. Thyri. *(PhD thesis) COLREGs-aware Trajectory Planning and Collision Avoidance for Autonomous Surface Vessels*. NTNU, 2022. URL https://hdl.handle.net/11250/3025463.

[37] Andreas B. Martinsen. *Optimization-based Planning and Control for Autonomous Surface Vehicles, Doctorial thesis*. NTNU, 2021.

[38] Edmund F. Brekke, Egil Eide, Bjørn-Olav H. Eriksen, Erik F. Wilthil, Morten Breivik, Even Skjellaug, Øystein K. Helgesen, Anastasios M. Lekkas, Andreas B. Martinsen, Emil H. Thyri, Tobias Torben, Erik Veitch, Ole A. Alsos, and Tor A. Johansen. *milliAmpere: An Autonomous Ferry Prototype*, volume 2311. IOP Publishing, 2022. URL https://dx.doi.org/10.1088/1742-6596/2311/1/012029.

[39] Kristoffer Bergman, Oskar Ljungqvist, Jonas Linder, and Daniel Axehill. *An Optimization-Based Motion Planner for Autonomous Maneuvering of Marine Vessels in Complex Environments.* 2020. doi: 10.48550/arXiv.2005.02674.

[40] Robin Deits and Russ Tedrake. *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*, volume 107. 04 2015. doi: 10.1007/978-3-319-16595-0_7.

[41] Sikang Liu, Michael Watterson, Kartik Mohta, Ke Sun, Subhrajit Bhattacharya, Camillo J. Taylor, and Vijay Kumar. *Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments*, volume 2. 2017. doi: 10.1109/LRA.2017.2663526.

[42] International Maritime Organization. *Convention on the International Regulations for Preventing Collisions at Sea*. 1972. URL http://inoa.net/zeilen/colreg.html.

[43] Emil H. Thyri, Erlend A. Basso, Morten Breivik, Kristin Y. Pettersen, Roger Skjetne, and Anastasios M. Lekkas. *Reactive collision avoidance for ASVs based on control barrier functions*. 2020. doi: 10.1109/CCTA41146.2020.9206340.

[44] Yoshiaki Kuwata, Michael T. Wolf, Dimitri Zarzhitsky, and Terrance L. Huntsberger. *Safe Maritime Autonomous Navigation With COLREGS, Using Velocity Obstacles*, volume 39. 2014. doi: 10.1109/JOE.2013.2254214.

[45] Mauro Candeloro, Anastasios Lekkas, and Asgeir Sørensen. *A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels*. Control Engineering Practice, 2017. doi: 10.1016/j.conengprac.2017.01.007.

[46] Steven M. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.

[47] Tim Zeitz. *NP-hardness of shortest path problems in networks with non-FIFO time-dependent travel times*, volume 179. Information Processing Letters, 2023. URL https://www.sciencedirect.com/science/article/pii/S0020019022000448.

[48] Harshita Sharma, Alexander Alekseychuk, Peter Leskovsky, Olaf Hellwich, R. S. Anand, Norman Zerbe, and Peter Hufnagl. *Determining similarity in histological images using graph-theoretic description and matching methods for content-based image retrieval in medical diagnostics*, volume 7. Diagnostic Pathology, 2012. doi: 10.1186/1746-1596-7-134.

[49] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd edition)*. Prentice Hall, 2010.

[50] Morten Breivik and Thor I. Fossen. Guidance laws for autonomous underwater vehicles. In Alexander V. Inzartsev, editor, *Underwater Vehicles*, chapter 4. IntechOpen, Rijeka, 2009. doi: 10.5772/6696. URL https://doi.org/10.5772/6696.

[51] L. E. Dubins. *On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents*, volume 79. American Journal of Mathematics, 1957. URL http://www.jstor.org/stable/2372560.

[52] Morten Breivik and Thor I. Fossen. *Applying Missile Guidance Concepts to Motion Control of MArine Craft*, volume 40. IFAC Proceedings Volumes, 2007. URL https://www.sciencedirect.com/science/article/pii/S1474667015321200. 7th IFAC Conference on Control Applications in Marine Systems.

[53] Stephen P. Bradley, Arnoldo C. Hax, and Thomas L. Magnanti. *Applied Mathematical Programming*. Addison-Wesley, 1977.

[54] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[55] C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder. *Efficient direct multiple shooting for nonlinear model predictive control on long horizons*, volume 22. Journal of Process Control, 2012. doi: https://doi.org/10.1016/j.jprocont.2012.01.008.

[56] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957. (Downloaded: 13.12.2022).

[57] M. Diehl, H.G. Bock, H. Diedam, and P.-B. Wieber. *Fast Direct Multiple Shooting Algorithms for Optimal Robot Control*, pages 65–93. Springer Berlin Heidelberg, 2006. URL https://doi.org/10.1007/978-3-540-36119-0_4.

[58] R. V. Gamkrelidze. *Discovery of the Maximum Principle*, volume 5. Journal of Dynamical and Control Systems, 1999. doi: 10.1023/A:1021783020548.

[59] Thomas J. Böhme and Benjamin Frank. *Indirect Methods for Optimal Control*. Springer International Publishing, 2017.

[60] Benjamin Passenberg. *Theory and Algorithms for Indirect Methods in Optimal Control of Hybrid Systems*. 2012.

[61] G. Hicks and W. Ray. *Approximation methods for optimal control synthesis*, pages 522–528. The Canadian Journal of Chemical Engineering, 1971.

[62] P. Deuflhard. *A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with application to multiple shooting*, pages 289–315. Numerische Mathematik, 1974.

[63] T. H. Tsang, D. M. Himmelblau, and T. F. Edgar. *Optimal control via collocation and non-linear programming*, volume 21, pages 763–768. Taylor & Francis, 1975.

[64] Matthew P. Kelly. *Transcription Methods for Trajectory Optimization: a beginners tutorial*. 2017. (Visited: 13.12.2022).

[65] Sébastien Gros and Moritz Diehl. *Numerical Optimal Control (Draft)*. 2022. (Visited: 13.12.2022).

[66] E. F. Camacho and C. Bordons. *Model Predictive control*. Springer London, 2007. doi: 10.1007/978-0-85729-398-5.

[67] Sean Gillies et al. Shapely: manipulation and analysis of geometric objects, 2007–. URL `https://github.com/Toblerity/Shapely`.

[68] Alan Bole, Alan Wall, and Andy Norris. Radar and arpa manual (third edition). pages 371–405. Butterworth-Heinemann, 2014.

[69] Øivind K. Kjerstad, Sveinung Løset, Roger Skjetne, and Runa A. Skarbø. *An Ice-Drift Estimation Algorithm Using Radar and Ship Motion Measurements*, volume 56. 2018.

[70] Stefan Kraemer, Christoph Stiller, and Mohamed Essayed Bouzouraa. *LiDAR-Based Object Tracking and Shape Estimation Using Polylines and Free-Space Information*. 2018.

[71] CheeKuang Tam and Richard Bucknall. *Collision risk assessment for ships*, volume 15. Journal of Marine Science and Technology, 09 2010. doi: 10.1007/s00773-010-0089-7.

[72] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, volume 4. 1968. doi: 10.1109/TSSC.1968.300136.

[73] Andreas B. Martinsen, Glenn Bitar, Anastasios M. Lekkas, and Sébastien Gros. *Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments*, volume 8. 2020. doi: 10.1109/ACCESS.2020.3037171.

[74] Andreas Wächter and Lorenz T. Biegler. *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*. Mathematical Programming, 2006.

[75] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi - A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, (11):1–36, 2019. doi: 10.1007/s12532-018-0139-4.

[76] Hongyang Zhu and Yi Ding. *Optimized Dynamic Collision Avoidance Algorithm for USV Path Planning*, volume 23. Sensors, 2023. doi: 10.3390/s23094567. URL `https://www.mdpi.com/1424-8220/23/9/4567`.

[77] Francisco Moreno-Mora, Lukas Beckenbach, and Stefan Streif. Predictive control with learning-based terminal costs using approximate value iteration, 2022.

[78] Johannes Köhler, Matthias A. Müller, and Frank Allgöwer. *A Nonlinear Model Predictive Control Framework Using Reference Generic Terminal Ingredients*, volume 65. 2020. doi: 10.1109/TAC.2019.2949350.

[79] Marco Gallieri and Jan M. Maciejowski. *Stabilising terminal cost and terminal controller for asso-MPC: enhanced optimality and region of attraction*. 2013 European Control Conference (ECC), 2013. doi: 10.23919/ECC.2013.6669549.

[80] Andreas B. Martinsen and Anastasios M. Lekkas. *Curved Path Following with Deep Reinforcement Learning: Results from Three Vessel Models*. 2018. doi: 10.1109/OCEANS.2018.8604829.

Appendix

A Vessel model

This thesis performed simulations on the research vessel milliAmpere. The vessel model is a simplified 3 DOF surge-coupled model and the parameters were retrieved from [9]. The model is presented in section 2.1 and given by:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\psi}\boldsymbol{\nu}, \tag{5.1a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}\boldsymbol{\nu} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{\tau}. \tag{5.1b}$$

The inertia matrix $\boldsymbol{M}$, Coriolis matrix $\boldsymbol{C}(\boldsymbol{\nu})$, and dampening matrix, $\boldsymbol{D}(\boldsymbol{\nu})$ and can be specified further as

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix}, \tag{5.2a}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & c_{13}(\boldsymbol{\nu}) \\ 0 & 0 & c_{23}(\boldsymbol{\nu}) \\ c_{31}(\boldsymbol{\nu}) & c_{32}(\boldsymbol{\nu}) & 0 \end{bmatrix}, \tag{5.2b}$$

$$\boldsymbol{D}(\boldsymbol{\nu}) = \begin{bmatrix} d_{11}(\boldsymbol{\nu}) & 0 & 0 \\ 0 & d_{22}(\boldsymbol{\nu}) & d_{23}(\boldsymbol{\nu}) \\ 0 & d_{32}(\boldsymbol{\nu}) & d_{33}(\boldsymbol{\nu}) \end{bmatrix}, \tag{5.2c}$$

The vessel model is assumed to utilize two azimuth thrusters, one aft and one bow, as illustrated in 2.2. Resulting in the thruster configuration matrix

$$\boldsymbol{T}(\boldsymbol{\alpha}) = \begin{bmatrix} \cos\alpha_1 & \cos\alpha_2 \\ \sin\alpha_1 & \sin\alpha_2 \\ l_{x,1}\sin(\alpha_1) & l_{x,2}\sin(\alpha_2) \end{bmatrix} \tag{5.3}$$

B Simulation parameters

Parameter	Value	Parameter	Value
m_{11}	2389.657	X_u	-27.632
m_{22}	2533.911	$X_{\|u\|u}$	-11.064
m_{23}	62.386	X_{uuu}	-13.965
m_{32}	28.141	Y_v	-52.947
m_{33}	5068.910	$Y_{\|v\|v}$	-116.486
c_{13}	$-m_{22}v - m_{23}r$	Y_{vvv}	-24.313
c_{23}	$m_{11}u$	$Y_{\|r\|v}$	-1540.383
c_{31}	$-c_{13}$	Y_r	24.735
c_{32}	$-c_{23}$	$Y_{\|v\|r}$	572.141
d_{11}	$-X_u - X_{\|u\|u}\|u\| - X_{uuu}u^2$	$Y_{\|r\|r}$	-115.457
d_{22}	$-Y_v - Y_{\|v\|v}\|v\| - Y_{vvv}v^2$	N_v	3.524
d_{23}	$-Y_r - Y_{\|v\|r}\|v\| - Y_{\|r\|v}\|r\|$	$N_{\|v\|v}$	-0.832
d_{32}	$-N_v - N_{\|v\|v}\|v\| - N_{\|r\|v}\|r\|$	$N_{\|r\|v}$	336.827
d_{33}	$-N_r - N_{\|v\|r}\|v\| - N_{\|r\|r}\|r\| - N_{rrr}r^2$	N_r	-122.860
$N_{\|r\|r}$	-874.428	N_{rrr}	0
$N_{\|v\|r}$	-121.957	L	5
$l_{x,1}$	1.8	$l_{y,1}$	0
$l_{x,2}$	-1.8	$l_{y,2}$	0

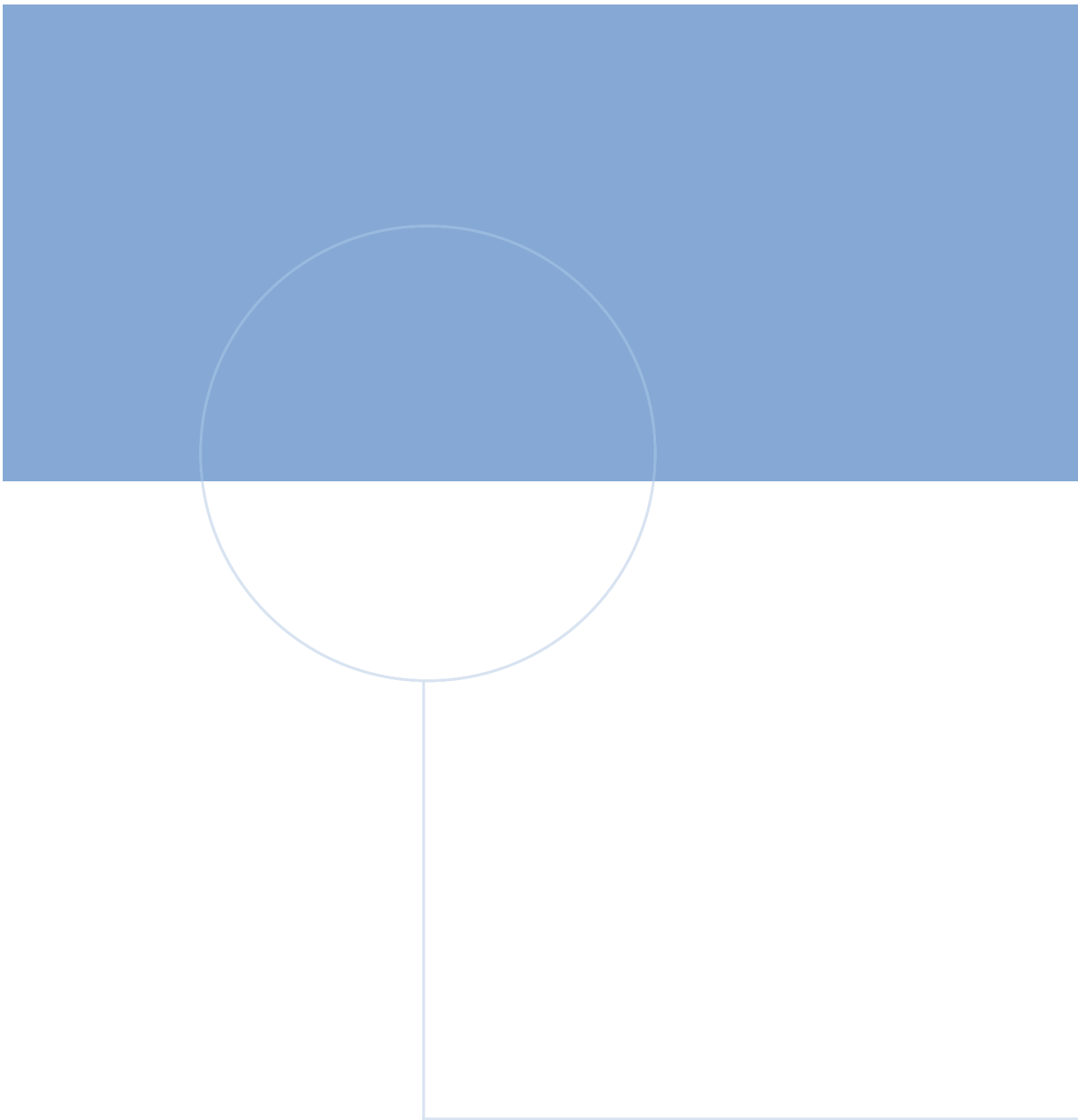
Table 5.1: milliAmpere model parameters.

Parameter	Value
$\boldsymbol{Q_\eta}$	$\begin{bmatrix} 35 & 0 & 0 \\ 0 & 35 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
$\boldsymbol{Q_\nu}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 100 \end{bmatrix}$
$\boldsymbol{R}_u$	$\begin{bmatrix} 1 \times 10^{-2} & 0 & 0 & 0 \\ 0 & 1 \times 10^{-2} & 0 & 0 \\ 0 & 0 & 1 \times 10^{-2} & 0 \\ 0 & 0 & 0 & 1 \times 10^{-2} \end{bmatrix}$
δ	10
σ	0
ϵ	1×10^{-3}
Δ_max	$2.5 \times L$
Δ_max	L
γ	0.1
y_{ew}	$\log(\sum_{i=2}^{N} y_e + 1)^2$
w	1×10^{-1}

Table 5.2: Optimization parameters for the vessel in transit.

Parameter	Value
$\boldsymbol{Q_\eta}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
$\boldsymbol{Q_\nu}$	$\begin{bmatrix} 150 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 100 \end{bmatrix}$
$\boldsymbol{R}_u$	$\begin{bmatrix} 1 \times 10^{-2} & 0 & 0 & 0 \\ 0 & 1 \times 10^{-2} & 0 & 0 \\ 0 & 0 & 1 \times 10^{-2} & 0 \\ 0 & 0 & 0 & 1 \times 10^{-2} \end{bmatrix}$
δ	40
σ	100
ϵ	1×10^{-3}
y_{ew}	0
w	1×10^{-1}

Table 5.3: Optimization parameters for the vessel when docking.



NTNU

Norwegian University of
Science and Technology