Master's thesis

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

Petter Knutsen Ødven

Static and dynamic multi-obstacle avoidance for docking of ASVs using computational geometry and numerical optimal control

Master's thesis in Cybernetics and Robotics
Supervisor: Anastasios Lekkas
June 2022

NTNU
Norwegian University of
Science and Technology

Petter Knutsen Ødven

Static and dynamic multi-obstacle avoidance for docking of ASVs using computational geometry and numerical optimal control

Master's thesis in Cybernetics and Robotics
Supervisor: Anastasios Lekkas
June 2022

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

Preface

This thesis concludes my work in the spring of 2022 for my Master of Technology. The thesis summarizes my findings on how to apply computational geometry and optimization theory for obstacle avoidance and docking of autonomous surface vessels. The work has been done at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology under the supervision of Anastasios Lekkas.

The recent years' advancements in autonomous docking of surface vehicles are my research's motivation. Promising results from optimization-based docking inspired me to investigate simultaneous obstacle avoidance and docking of an autonomous surface vessel (ASV). The thesis is a continuation of my project report (Ødven (2021)) and is greatly inspired by the work of Martinsen et al. (2019) and Bitar et al. (2019).

The main contribution of this thesis is docking an ASV while avoiding both static and dynamic obstacles. We propose two successful methodologies utilizing computational geometry and optimization theory to create feasible obstacle-free docking trajectories. The main tool was Python for programming and simulation purposes. Various open-source libraries have been used to ease the implementation, this includes NumPy (Oliphant (2021)), Matplotlib (matplotlib development team (2021)), SciPy (sciPy community (2021)) and Shapely (Gillies (2021)). The optimal control problem was implemented with the use of CasADi (Andersson et al. (2021)) with a linear solver from HSL (2022). Some relevant material from MSS toolbox (Fossen (2021)) were rewritten for Python.

I would like to thank my supervisor, Anastasios Lekkas, for his non-stop energy and engaging discussions. A lot of my work would not have come to life without his creativity and insight into marine autonomy. I would like to thank Andreas Bell Martinsen for co-authoring a paper during the spring and for his interesting work throughout the years which serves as a foundation for the implementations in this thesis. I would also like to thank my friends for our great lunches throughout the semester, keeping up the motivation. My partner, Emilie Hope, and my family for being supportive and helpful throughout my education.

02.06.2022

[illegible]

Abstract

Autonomous systems will likely have a significant impact on our next-generation modern society and replace humans in a variety of today's tasks. Automatic docking of autonomous surface vehicles (ASVs) is no different. Overcoming challenges like low-speed maneuvering, environmental forces, and traffic requires sophisticated systems. Optimization-based methods have shown promising results over the recent years. However, they are still limited when dealing with non-convex harbor configurations and obstacles, as this adds the significant challenge of non-convex constraints.

This thesis presents two methods, A and B, that successfully dock an ASV in real-time while avoiding static and dynamic obstacles obstructing the direct path to the quay. We propose a novel tangential decomposition of a non-convex harbor area into convex polygons for method A. The decomposition leads to the A* search finding the optimal obstacle-free sequence of polygons connecting the initial pose to the dock pose. Finally, a switch-based optimization problem computes a feasible, obstacle-free path to the dock. For method B, we propose a constrained Delaunay triangulation into a medial axis-based road map connecting the initial pose to the dock pose. Then, search and waypoint reducing algorithms compute an initial guess that warm-starts an optimization problem to find an obstacle-free trajectory.

From our results, computational geometry seems promising when pairing it with optimization-based approaches. We maintain the benefits of optimization-based methods with respect to vehicle kinematics and dynamics, actuator, and spatial constraints. Additionally, this combination helps our proposed methods avoid local optima and successfully dock in obstacle obstructed harbor environments.

Sammendrag

Autonome system vil sannsynligvis ha stor innvirkning på vårt neste-generasjons moderne samfunn og ersatte mennesker i en rekke av dagens oppgaver. Automatisk dokking av autonome overflatefartøy er intet unntak. Å overmanne utfordringer knyttet til lavhastighetsmanøvrering, miljøkrefter og trafikk krever sofistikerte systemer. Optimeringsbaserte metoder har vist lovende resultater de siste årene. Likevel, så er de fortsatt begrenset når det gjelder ikke-konvekse havneområder og hindringer, fordi dette gir ekstra utfordringer i form av ikke-konvekse bibetingelser.

I denne avhandlingen presenterer vi to metoder, A og B, som vellykket dokker et autonomt overflatefartøy i sanntid mens det unngår statiske og dynamiske hindringer som hindrer den naturlige stien til kaien. For metode A foreslår vi en ny tangentiell dekomponering av et ikke-konvekst havneområde til konvekse polygoner. Dekomponeringen fører til at A* algoritmen finner den optimale hindringsfrie sekvensen av polygoner som forbinder initialposisjonen til sluttposisjonen. Til slutt beregner et svitsjbasert optimeringsproblem en hindringsfri sti for å gjennomføre presis og sikker dokking. For metode B foreslår vi en begrenset Delaunay triangulering som leder til et veikart basert på et topologisk skjelett som forbinder initialposisjonen med sluttposisjonen. Deretter beregner en søk- og veipunktsreduserende algoritme et initialgjett til et optimeringsproblem som finner en hindringsfri sti til sluttposisjonen.

Ut fra resultatene i avhandlingen virker beregningsgeometri lovende når det kombineres med optimeringsbaserte tilnærminger. Kombinasjonen lar oss opprettholde fordelene med optimeringsbaserte tilnærminger med hensyn til fartøyskinematikk og dynamikk, samt aktuator og romlige bibetingelser. I tillegg hjelper denne kombinasjonen våre foreslåtte metoder med å unngå lokale optima og dermed lykkes med å legge til kai i obstruerte havnemiljøer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and motivation

Autonomy will shape our future of technology and most likely replace humans in many of today's tasks. When we talk about autonomous systems in terms of technology, we usually refer to systems manifesting self-governing behavior without the need for human interaction to complete the task it has been specified to do. Industrial robots could, for instance, entirely transform the logistics, safety, and efficiency of the industry. Pin-point accurate surgical robots can execute complex surgeries and medical procedures (Leonard et al. (2014)). Severely reducing the risk of missteps and possibly opening up new procedural possibilities that formerly were considered too dangerous to be performed by a human being. Already, we can see the autonomous car industry, led by Tesla, providing systems for the next generation of driver-less cars (Tesla (2022)).

Marine autonomy is a growing field both academically and industrially. Most research focuses on underwater and surface vehicles, and the use cases are vast. Autonomous underwater vehicles are applicable for deep-water exploration, offshore maintenance, and data collection. Autonomous surface vehicles (ASVs) are often related to cargo or passenger transportation, remote maintenance operations, and surveillance.

We can generally break down an ASV operation into three phases. The undocking, transit, and docking phase. Much research has been conducted on the large, open-water transit phase of an ASV operation. Such operations usually mean more open space to maneuver in and less traffic. The requirements for a system handling a transit are not as sophisticated as in a docking situation. In order to form a completely autonomous system, docking is an equally important part of a short-ranged ASV operation as the transit. Complex requirements might be why docking-related operations have not seen the same attention until recently.

Docking usually refers to the process of mooring a vessel to a quay or similar structure. In the literature, docking usually refers to the movement from open waters to a stationary position along the quay. Small confined harbors paired with moving traffic and low-speed maneuvering create complex challenges for an autonomous vessel. We need robust and accurate systems to account for unmodeled vessel dynamics, which are more prevalent when maneuvering at low speeds, and traffic-related challenges.

The harbor is usually the area where a vessel encounters the most amount of traffic. Thus, sophisticated situational awareness and collision avoidance systems are a necessity. Solving docking-related challenges can lead to benefits like increased energy optimization, improved safety, and reduced operational costs. Additionally, it will be one step closer to a fully autonomous system.

Until recently, research on docking has been scarce. Early work used fuzzy logic control in Rae et al. (1993), artificial neural networks (ANN) in Yamato (1990), and Shuai et al. (2019). Breivik and Loberg (2011) proposed a two-staged virtual target-based docking procedure. The interest in automatic docking was seriously set in motion when Martinsen et al. (2019) demonstrated promising results using nonlinear model predictive control (NMPC) combined with convex spatial constraints taking the vessel's hull into account for collision avoidance. Their work led to the continuation of Bitar et al. (2020), where they used a high-level optimization planner paired with a low-level DP controller to do full-scale docking. Martinsen et al. (2020) extended this work even further by accounting for map inaccuracies using exteroceptive sensors and dynamically updating the convex spatial constraints.

Other, more recent work includes Bergman et al. (2020). They apply a lattice-based motion planner to warm-start a receding horizon optimization problem. They also incorporate a safety envelope around the vessel, which is a similar approach to the convex spatial constraints first presented in Martinsen et al. (2019). Another recent research, like the study by Li et al. (2020) uses a multi-objective optimization strategy to design an NMPC controller. Their proposed solution eliminates the need to tune the objective function parameters and outperforms a regular NMPC controller. Already existing optimization-based techniques on transit can, for instance, be applied for docking an ASV. In the work of Bitar et al. (2019), they use a warm-started optimization-based trajectory planner to solve a transit phase going from A to B while simultaneously avoiding obstacles. Large parts of this thesis can be viewed as a continuation and adaptation of their work into a docking scenario.

Other techniques apart from optimization-based methods have also been investigated. In the work of Sawada et al. (2020), they propose a novel path-following algorithm and speed controller to perform the docking maneuver. Their proposed solution handles the computational expenses and non-linearity better than optimization-based methods. Although some studies have been done, more sophisticated control approaches are not widespread. In the work of Baek and Woo (2022), they use a model reference adaptive controller showing promising results when dealing with environmental forces in a docking scenario. Other work investigating more sophisticated control includes the work of Torvund (2020). They investigate non-linear autonomous docking and path-following control for an underactuated vessel. They compared a proportional-integral-derivative (PID) controller, a PID sliding mode controller (PID-SMC), and a super-twisting controller (STC). Their results indicate that more advanced control approaches could benefit path-following purposes. However, their work focuses more on obstacle avoidance and path-following cross-track error, not on positional accuracy when doing low-speed maneuvering to reach a stationary quay position.

In the field of deep learning, Hammervold (2020) and Strand (2020) have done extensive research on the proximal policy optimization (PPO) and the deep deterministic policy gradient (DDPG) algorithms. They accomplished successful docking in simulation using both algorithms. However, their studies are limited to achieving successful low-speed docking in unobstructed harbors and do not account for obstacle avoidance. Work from Cai et al. (2021) investigates a hybrid RL-optimization-based trajectory planning and obstacle avoidance to solve both transit and docking, connecting two of the phases. Such hybrid control approaches seem promising for future control purposes, despite their work not considering obstacle and collision avoidance for docking in small confined harbor spaces. Bitar et al. (2021)'s work shows how to connect the three phases of an ASV operation by conditionally switching between different planning and control approaches.

An issue with many existing docking studies is that they are limited to dynamic positioning scenarios. They do not consider docking as the event of achieving a stationary position alongside the quay. When the vessel has to react to the quay and avoid colliding with it, we limit the vessel's maneuverability in its approach to the quay, which makes a considerably more realistic and challenging docking scenario. Most optimization-based methods are limited by not having a guarantee for finding the global optima. In many cases, local optima would still be sufficient. E.g., in an open-water transit path, the probability of finding a local optimum resulting in an infeasible solution is not as high as in narrow space maneuvering like a docking scenario. Thus, local optima might not be sufficient when docking with traffic in a narrow harbor area.

A few studies, like Martinsen et al. (2019), Bitar et al. (2020), Martinsen et al. (2020), and Bergman et al. (2020), use convex spatial constraints in the optimization for collision avoidance. Their approaches are prone to get stuck in local optima next to obstacles and never reach the dock pose. Some more recent approaches address this challenge. In work by MIY (2022), they successfully use optimal control when docking a vessel in both complex harbor geometry. A significant limitation to their solution is the computational expenses where the optimization problem takes a few days to complete. A continuation of this work led to a warm-started solution in Rachman et al. (2022), dealing with the real-time feasibility challenges from MIY (2022). They developed the collision avoidance algorithm to handle both convex and non-convex harbor configurations. However, none of the methods address challenges related to polygonal holes or avoiding the local optima created by obstacles blocking the path to the dock pose. In the work of Zhou et al. (2022), they utilize optimization-based techniques to dock a formation of ASVs. Their approach to collision avoidance is in terms of the formation not colliding with each other, which shows promising results for more traffic-related challenges dealing with other moving vessels and cases where united control of an entire fleet is necessary.

1.2 Goal and research questions

This thesis investigates how we can successfully dock while avoiding multiple obstacles, both static and dynamic, using computational geometry and optimization theory for collision avoidance and guidance purposes. An essential prerequisite for our approach is the utilization of convex constraints as a safety guarantee for the vessel.

From this, the following research questions are formulated:

- How can we extend the work from Ødven (2021) to avoid multiple obstacles, both static and dynamic, and successfully dock a vessel?

- Can we use other computational geometry concepts and existing techniques to avoid multiple obstacles, both static and dynamic, and successfully dock a vessel?

1.3 Contributions

This thesis presents two methods (A and B) that successfully dock a vessel while avoiding static and dynamic obstacles obstructing the vessel's path to the dock pose. Method A is first and foremost an extension of the work from Martinsen et al. (2019) and a continuation of the work done in the project report (Ødven (2021)). Bitar et al. (2019) heavily inspired our proposed method B. Our methodologies are primarily a combination of computational geometry, optimization theory, and low-level motion control.

The contributions of this thesis are as follows:

- We propose two methodologies that successfully dock while avoiding static and dynamic obstacles. The methods account for newly discovered obstacles and can replan and react to previously unknown obstacles.
- We have further developed the method proposed by Ødven (2021). We have improved the decomposition to incorporate more of the geometric shapes that emerge from tangential decomposition. Additionally, we implemented an improved switching mechanism to speed up the docking procedure for closed-loop control.
- For method B, we extend the work from Bitar et al. (2019). The continuation incorporates constrained Delaunay triangulation decomposition and medial axis to determine a straight-lined initial guess to warm-start an optimization-based trajectory generator.
- Both methods incorporate a separated docking control system, with the methodologies used as high-level optimization planners combined with a low-level DP controller.
- A paper on static multi-obstacle avoidance for docking of an ASV (Ødven et al. (2022)), accepted for the 14th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles.

1.4 Thesis structure

The rest of the thesis is organized as follows. Chapter 2 presents the relevant theoretical background and concepts relevant to the work presented in this thesis. The chapter introduces computational geometry, marine maneuvering models, motion control, and optimization theory. Chapter 3 presents a detailed explanation of our two proposed methodologies for automatic docking of an ASV. We thoroughly explain how we combine computational geometry, optimization theory, and motion control into a system able to avoid obstacles and dock an ASV. Chapter 4 presents simulation results for different scenarios, and Chapter 5 concludes the thesis.

Chapter 2

Theoretical background

This chapter presents the necessary theoretical background to form a basis for the rest of the thesis. We introduce concepts related to computational geometry, motion control, path planning, and optimization theory. This thesis is a continuation of the work in the project report (Ødven (2021)). Thus, they are built on much of the same theoretical foundation.

2.1 Computational geometry

Computational geometry is a mathematical field focused on geometric shapes and their properties in designing and implementing geometric algorithms. The following section presents the relevant theory related to convex polygons, polygon decomposition, triangulation, and medial axis and how this forms a basis for the thesis's path planning and obstacle avoidance.

2.1.1 Convex polygon

A large part of this thesis builds on properties of convexity, more specifically, the properties of convex polygons. They are advantageous because we can directly implement them as constraints in optimization problems.

To define a convex polygon, we first define a convex set $\boldsymbol{C}$ consisting of points $\boldsymbol{c}_i$

where $i = 1, ..., n$ and n is the number of points. For $\boldsymbol{C}$ to be convex, any line segment $l_{i,j}$ between two points $\boldsymbol{c}_i, \boldsymbol{c}_j$ has to be a subset of $\boldsymbol{C}$.

From our definition of a convex set, we can define a convex polygon as the polygon enclosing the convex set $\boldsymbol{C}$. Let us define a set of straight-line segments $\boldsymbol{L}$ connecting the points $\boldsymbol{c}_i$. See Figure 2.1.1, where the line segments $\boldsymbol{l}_{i,j}$ forms a convex polygon $\boldsymbol{P}$.

We can describe $\boldsymbol{P}$ as the solution to the linear inequalities,

$$\boldsymbol{A}\boldsymbol{c}_i \leq \boldsymbol{b}, \quad \text{where } i = 1, ..., s, \tag{2.1.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{s \times 2}$, $\boldsymbol{b} \in \mathbb{R}^{s \times 1}$, $\boldsymbol{c}_i \in \mathbb{R}^2$ is vertex i's Cartesian position (x, y), and s is the number of vertices in $\boldsymbol{P}$. By formulating $\boldsymbol{P}$ on the linear inequality form (2.1.1), we can directly include it as an inequality constraint in an optimization problem.

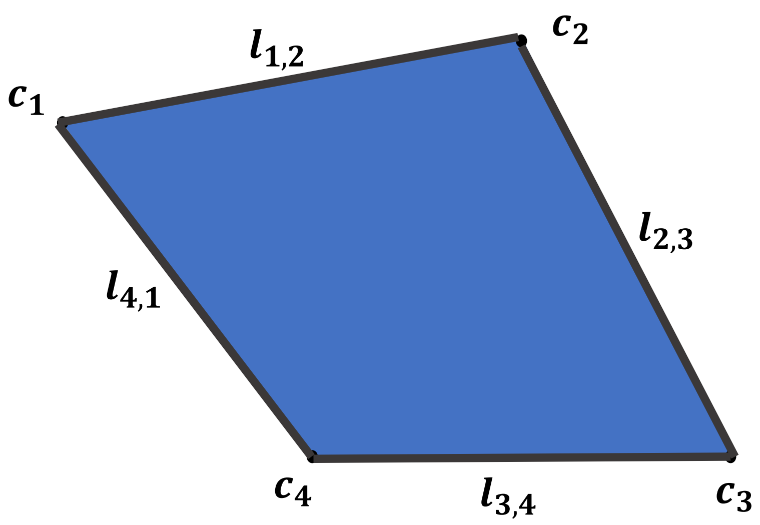


Figure 2.1.1: A convex polygon with vertices $\boldsymbol{c}_i$ for $i = 0, .., 4$ and line segments $l_{i,j}$

2.1.2 Polygon decomposition

Polygon decomposition is related to the partition and separation of a polygon into a set of simpler geometric shapes. The term is a more general term than the broader known polygon partitioning problem, in which it allows for overlapping subsets. As we want to capitalize on the convexity of polygons, a decomposition into convex geometric shapes becomes applicable when dealing with non-convex polygons.

Several methods for decomposing polygons exist in the field of computational geometry. Triangulation, for instance, serves a purpose where algorithms are usually fast and straightforward with a guarantee of convexity in terms of triangles. While more complex algorithms, like the minimum convex partitioning, are usually more complicated to design.

2.1.3 Delaunay triangulation

Delaunay triangulation (DT) is a computational geometry concept based on the circumcircles of a set of points. To define Delaunay triangulation, we first define a set of points $\boldsymbol{S}$ in the plane. The set $\boldsymbol{S}$ lets us define a Delaunay triangulation $DT(\boldsymbol{S})$ as the triangulation T of $\boldsymbol{S}$ such that no point in $\boldsymbol{S}$ is inside the circumcircle of any triangle of $DT(\boldsymbol{S})$. A Delaunay triangulation does not take any edges into account when triangulating, only the set of points. Without accounting for edges, we run into difficulties when dealing with holes in a polygon, where it is necessary to triangulate based on specific constraints.

Lee (1978) was the first to describe constrained Delaunay triangulation (CDT) as a generalization of the Delaunay triangulation problem. To define CDT, we first define a straight-line planar graph $\boldsymbol{G}$ with the same set of points $\boldsymbol{S}$. The constrained Delaunay triangulation $CDT(\boldsymbol{G})$ is such that every edge of $\boldsymbol{G}$ has to be an edge of $CDT(\boldsymbol{G})$. For the remaining points, no point can be inside the circumcircle of any triangle of $CDT(\boldsymbol{G})$ except when the circumcircle crosses an edge of $\boldsymbol{G}$. From the definition; it follows that if $\boldsymbol{G}$ does not have any edges, $CDT(\boldsymbol{G}) = DT(\boldsymbol{S})$. However, as long as it exist an edge in $\boldsymbol{G}$, the solution of $CDT(\boldsymbol{G})$ will be as accurate as possible to the solution of $DT(\boldsymbol{S})$. We can effectively triangulate a polygon with holes by utilizing CDT as it takes edges into account, as shown in Figure 2.1.2.

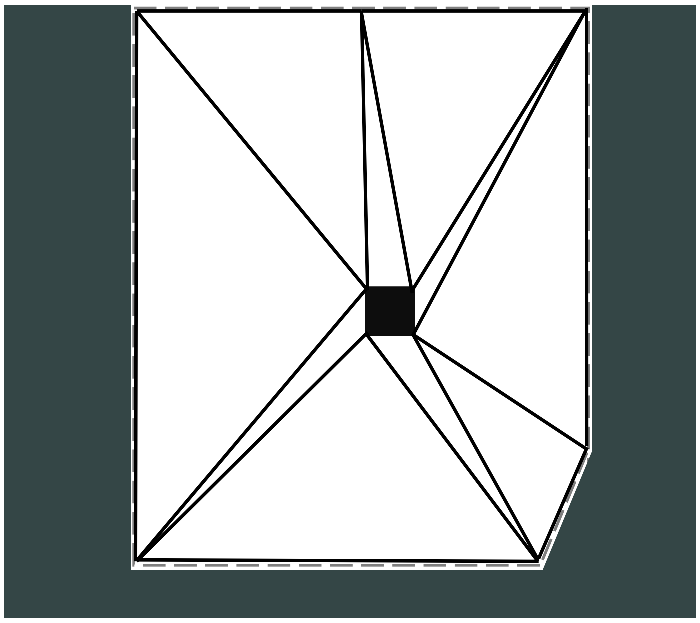


Figure 2.1.2: Constrained Delaunay decomposition of a polygon $\boldsymbol{P}$ (grey stapled line) with an obstacle as a hole

2.1.4 Voronoi diagram and medial axis

The Voronoi diagram is the dual problem of a Delaunay triangulation. Contrary to triangulating a set of points $\boldsymbol{S}$, a Voronoi diagram is a partition of space based on equidistance. Hence, they are closely connected. In order to define the Voronoi diagram, let us define a set of points $\boldsymbol{C}$ in the plane. The Voronoi region $\boldsymbol{R}$ of a point $c \in \boldsymbol{C}$ is the set of points as close to c as any other point in $\boldsymbol{C}$. The Voronoi diagram is then, $\boldsymbol{V}(C) = \bigcup \boldsymbol{R}_i$ for $i = 1, .., n$, where n is the number of points in the set $\boldsymbol{C}$. Figure 2.1.3 shows a simple example of a Voronoi diagram and its relation to the dual problem, Delaunay triangulation.

A closely related problem of the Voronoi diagram is the medial axis (MA). MA is the locus of centers of locally maximal circles inside an object (Chang (2013)). The medial axis is a subset of the Voronoi diagram of the edges and vertices of the polygon. For this thesis, we define the medial axis as the edges connecting the centroids of the triangles, as shown in Figure 2.1.4. The resulting medial axis is suboptimal because the CDT is not a true Delaunay triangulation.

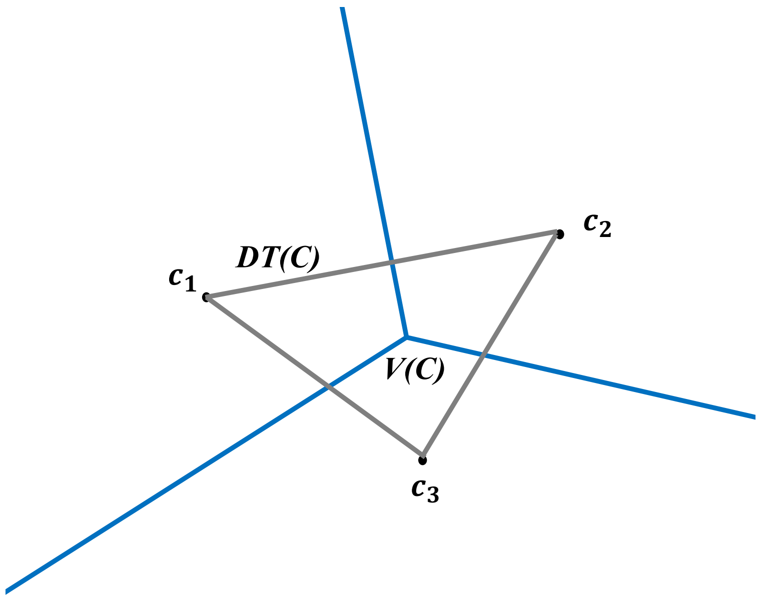


Figure 2.1.3: A Voronoi diagram and its relation to the Delaunay triangulation

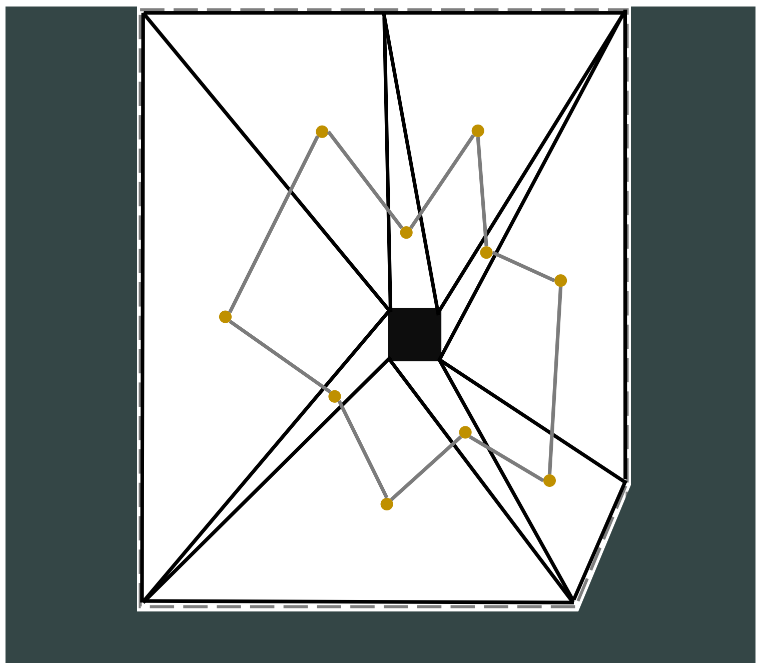


Figure 2.1.4: Medial axis formed by the edges connecting the triangles centriod

2.2 Vessel model

In order to plan and simulate a docking trajectory for an ASV using optimization theory, we need to take the vessel's kinematics and dynamics into account. This section describes a marine vessel's kinematics, dynamics, and thrust configuration.

2.2.1 Vessel kinematics and dynamics

We define a vessel's kinematics and dynamics using a 3 DOF maneuvering model from Fossen (2011), as shown in Figure 2.2.1. We denote the pose vector $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times \mathbb{S}$ as the Cartesian position (x, y) in a North-East-Down (NED) reference frame and the yaw angle ψ of the vessel. The velocity vector $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$ is the surge u, sway v, and yaw rate r. The generalized control forces $\boldsymbol{\tau} = [f_u, f_v, f_r]^\top \in \mathbb{R}^3$ is the force and moment in the surge, sway, and yaw. The model kinematics and dynamics are as follows,

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu}, \tag{2.2.1}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \tag{2.2.2}$$

where the inertia matrix $\boldsymbol{M} \in \mathbb{R}^{3\times3}$, the Coriolis matrix $\boldsymbol{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, the dampening matrix $\boldsymbol{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$. The rotation matrix $\boldsymbol{R} \in SO(3)$ is given by,

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2.3}$$

2.2.2 Thrust configuration

The thrust configuration lets us map the generalized control forces $\boldsymbol{\tau}$ to the thrust forces $\boldsymbol{f}$ as Fossen (2011) formulated,

$$\boldsymbol{\tau} = \boldsymbol{T}(\boldsymbol{\alpha})\,\boldsymbol{f}, \tag{2.2.4}$$

where the thrust configuration matrix $\boldsymbol{T}(\boldsymbol{\alpha}) \in \mathbb{R}^{3\times n}$, and the thrust force $\boldsymbol{f} \in \mathbb{R}^n$. The thrust configuration matrix for n thrusters is as follows,

$$\boldsymbol{T}_i(\boldsymbol{\alpha}) = \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \\ L_{x,i}\,\sin(\alpha_i) - L_{y,i}\,\cos(\alpha_i) \end{bmatrix}, \tag{2.2.5}$$

where $i = 1, ..., n$, $L_{x,i}$ and $L_{y,i}$ is thruster i's position in x and y with respect to the body-fixed CO. The thrust angles $\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 & \dots & \alpha_n\end{bmatrix}^\top$ represents the thruster force angle with respect to the body-fixed surge vector u_i of the thruster. Figure 2.2.2 shows a thrust configuration for a vessel with two thrusters.

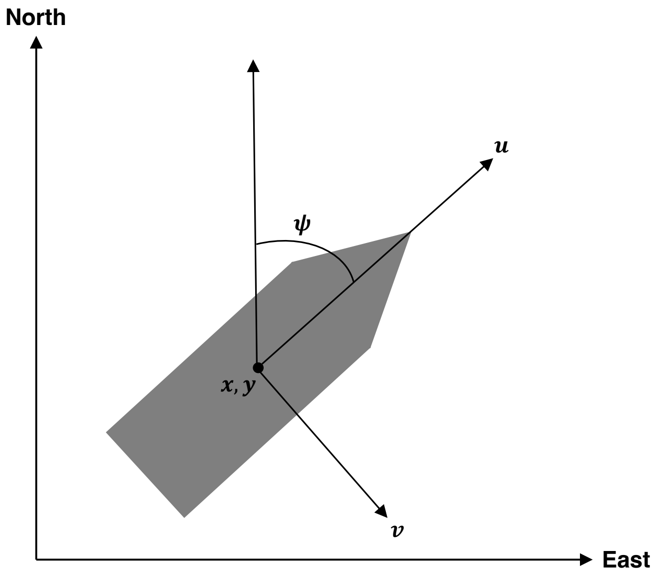


Figure 2.2.1: 3 DOF surge, sway, yaw vessel model

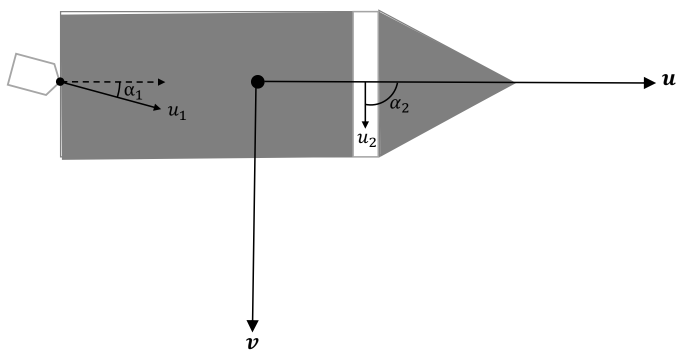


Figure 2.2.2: A vessel's thruster configuration for two thrusters. One azimuth thruster in the aft of the ship, and one tunnel thruster in the bow

2.3 Motion control

Motion control in marine systems is a vast topic often referred to as guidance, navigation, and control (GNC) (Fossen (2011)). This thesis will focus on high-level waypoint-based path planning for trajectory generation as guidance systems. Our simulations will have both open-loop optimal control and closed-loop dynamic positioning control as our low-level motion control. This section provides theoretical background of path planning and low-level motion control concepts.

2.3.1 Path planning

In its most general term, we can describe path planning as reaching a specific goal state. In terms of marine guidance and motion control, path planning usually refers to reaching a specific positional state while reacting to the environment. Figure 2.3.1 shows a typical case of path planning, which is to find the shortest collision-free path from an initial state to a goal state. Several different optimality criteria exist, where the shortest path is one of the most common. Other criteria could be traveling in the shortest amount of time, energy efficiency, and account for model dynamics.

The vast opportunities make path planning a complex topic where one can tailor solutions to suit various needs. There are, however, some distinctions that allow us to classify path planning algorithms. In Figure 2.3.1, the environment is static and fully known, which generally means that nothing in the environment except for the agent can move. Additionally, we have complete information of the map. The opposite case is when the environment is dynamic and partially known. Such a case usually means that the agent has no prior information and that it needs to rely on sensors to discover and build the map while simultaneously planning a collision-free path. These two scenarios have very different requirements for computational costs and the speed of the algorithms. A more computationally efficient algorithm could be more suitable for a large environment, whereas an accurate algorithm could be more suitable in smaller complex environments.

In order to serve various requirements, numerous different methods have arisen. Optimization-based techniques could, for instance, find a path satisfying several optimality criteria simultaneously. Other techniques such as the versatile search algorithms, Rapidly-Exploring Random Trees, efficiently exploring high-dimensional

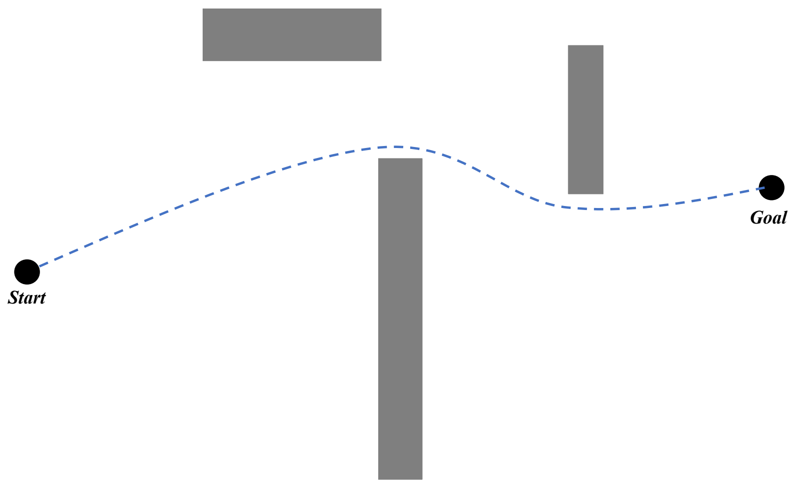


Figure 2.3.1: Collision free trajectory in a static, fully known environment

space (LaValle (1998)), and the well-known Djikstra and A* star algorithms, find the shortest path in a discrete, fully known static environment.

The path planning in this thesis will apply various techniques suited to our needs. We will apply computational geometry to create a decomposed discrete space where the A* can find the shortest path. Then, optimization-based techniques will further refine this path by satisfying various constraints.

Waypoint-based path planning

A common approach in marine guidance is using waypoints for path planning. They are used to indicate a change in the path, usually a directional change, but it could also be related to speed. A frequent definition of a waypoint is a Cartesian position (x, y). However, in this thesis, we define it as a pose, $\boldsymbol{\eta}_{w,i} = [x_{w,i}, y_{w,i}, \psi_{w,i}]^\top$, which incorporate the heading angle ψ of the vessel. Generating a series of waypoints allows us to construct a series of valid positional states to reach a goal state and incorporate guidance laws and low-level motion control to achieve such a state. Figure 2.3.2 shows how we can solve the scenario in Figure 2.3.1 by generating a set of waypoints to track.

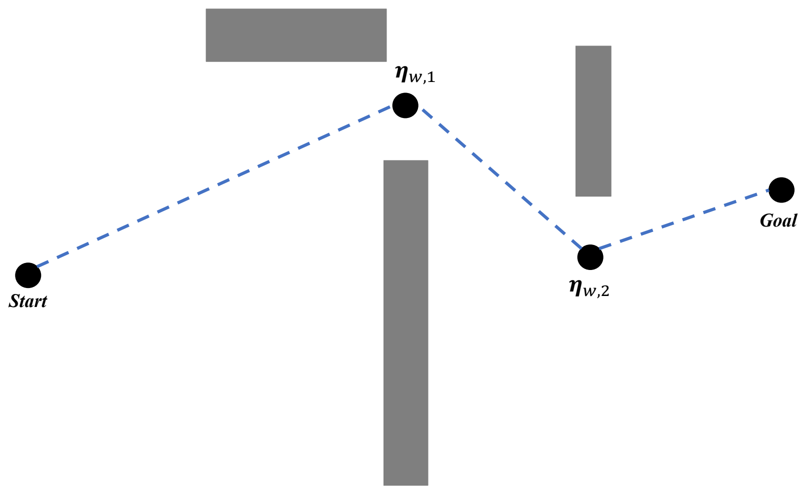


Figure 2.3.2: Waypoint-based collision-free trajectory in a static, fully known environment

A* search algorithm

The A* search algorithm is, in computer science, a graph traversal algorithm. The algorithm is recognized as an extension of the Djikstra algorithm by using heuristics to improve the performance. The A* is a best-first algorithm and determines the node to explore by choosing the lowest estimated cost to the goal node. If we define a graph $\boldsymbol{G}$, the A* algorithm explores each node in $\boldsymbol{G}$ and finds the shortest path from start to goal.

Let us define the travel cost from the current node c to a node n, as $g(n)$. We define the heuristic $h(n)$ as the estimate of the cost from node n to the goal node. When exploring, the A* algorithm chooses the next node n to visit by minimizing the cost $f(n) = g(n) + h(n)$. The A* is well known for its optimality and speed, and as long as the heuristic $h(n)$ does not overestimate the travel cost to the goal node, A* guarantees optimality. I.e., it guarantees to find the shortest path from start to goal. Additionally, it guarantees completeness, which means that as long as a path exists from start to goal, the A* finds the shortest.

The time complexity $O(b^{\epsilon d})$ is exponential to the depth d of the solution. We denote the branching factor as b and the relative heuristic error as $\epsilon = \frac{h* - h}{h*}$, where $h*$ is the optimal heuristic (Russell and Norvig (2010)). The worst-case complexity for both time and space is then $O(b^d)$. The space complexity is the main drawback of the A* because it keeps all generated nodes in memory (Russell and Norvig (2010)).

Algorithm 1 from Sharma et al. (2012) shows the pseudocode for the A* algorithm.

Algorithm 1: A* search algorithm

```
initialize an empty open and closed list;
append starting node to open;
while open is not empty do
    current $\leftarrow$ node with lowest f();
    pop current from open;
    append current to closed;
    if current $=$ goal then
        return;
    for each neighbor of current do
        if neighbor $\in$ closed then
            continue;
        if neighbor $\in$ open and g(current)+distance(current,neighbor) $<$ g(neigbor) then
            pop neighbor from open;
        if neighbor $\in$ closed and g(current)+distance(current,neighbor) $<$ g(neigbor) then
            pop neighbor from closed;
        if neighbor $\notin$ open and $\notin$ closed then
            append neighbor to open;
            g(neighbor) $\leftarrow$ g(current)+distance(current,neighbor);
            h(neighbor) $\leftarrow$ heuristic to goal;
            f(neighbor) $\leftarrow$ g(neighbor) + h(neighbor);
return failure;
```

2.3.2 Trajectory tracking

Trajectory tracking refers to the objective of forcing a system output $\boldsymbol{y}(t) \in \mathbb{R}^m$ to track the desired output $\boldsymbol{y}_d(t) \in \mathbb{R}^m$ (Fossen (2011)). In the context of marine guidance, we usually separate between path and trajectory. A path usually refers to a time-independent path, and a trajectory refers to a time-dependent path. Hence, trajectory tracking is normally a more demanding task for low-level control to track in comparison to a time-independent path. In order to track the generated time-varying reference $\boldsymbol{\eta}_d = [x_d, y_d, \psi_d]^\top \in \mathbb{R}^2 \times \mathbb{S}$, we need to minimize the tracking error,

$$\boldsymbol{\eta} - \boldsymbol{\eta}_d = \begin{bmatrix} x - x_d \\ y - y_d \\ \psi - \psi_d \end{bmatrix}. \tag{2.3.1}$$

A common approach in marine guidance is to use path-following methods. Path-following is more common than trajectory tracking because trajectory tracking sets requirements for the vessel's position with respect to time. Thus, increasing the complexity of low-level motion control because of unmodeled dynamics. Implementing time-varying paths, especially generating them with optimization, could be beneficial because it allows us to incorporate constraints in the path generation, e.g., the optimality criteria mentioned in Section 2.3.1. This thesis uses optimization and constraints to account for vessel dynamics and obstacles when generating a reference trajectory for a low-level controller or as the direct control input to a vessel.

2.3.3 Dynamic positioning systems

Dynamic positioning (DP) usually means controlling a vessel in low-speed situations and is often used for station-keeping. Environmental forces can considerably impact the vessel with low speeds, and a DP controller must be robust and compensate for such unmodeled dynamics. Because of this, a DP controller is advantageous for docking situations where most of the vessel maneuvering is at low speeds in a harbor area. Fossen (2011) describes a DP controller as,

$$\boldsymbol{\tau} \; = \; -\hat{\boldsymbol{\tau}}_{wind} \; + \; \boldsymbol{R}^\top(\psi)\boldsymbol{\tau}_{PID}, \tag{2.3.2}$$

where $\hat{\boldsymbol{\tau}}_{wind}$ is an estimate of the wind forces, and $\boldsymbol{\tau}_{PID}$ is a proportional-integral-derivative (PID) controller.

By neglecting the environmental forces and adding a feed-forward term to account for unmodeled dynamics, we describe the trajectory tracking DP controller from Bitar et al. (2020) as,

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{ff} + \boldsymbol{R}^\top(\psi)\boldsymbol{\tau}_{PID}, \tag{2.3.3}$$

where the feed-forward term is,

$$\boldsymbol{\tau}_{ff} \; = \; \boldsymbol{M}\dot{\boldsymbol{\nu}} \; + \; \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}, \tag{2.3.4}$$

and the PID controller,

$$\boldsymbol{\tau}_{PID} = -\boldsymbol{K}_p\tilde{\boldsymbol{\eta}} - \boldsymbol{K}_d\dot{\boldsymbol{\eta}} - \boldsymbol{K}_i\int_0^t \tilde{\boldsymbol{\eta}}(\tau)d\tau, \tag{2.3.5}$$

where $\boldsymbol{K}_p, \boldsymbol{K}_d$, and $\boldsymbol{K}_i$ are the proportional, derivative, and integral controller gain. We also have the state derivative $\dot{\boldsymbol{\eta}}$ and the state error $\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta} - \boldsymbol{\eta_d}$, where $\boldsymbol{\eta_d}$ is the desired state.

2.4 Optimal control

Optimal control theory is a branch of mathematical optimization developed to control dynamic systems while fulfilling optimality criteria formulated in an objective function and applicable in many industries, from economy to autonomous control of marine vessels. For this thesis, optimal control is the approach used for trajectory generation for a DP controller and open-loop control.

2.4.1 Dynamic systems

In general, we can explain a dynamic system as a representation of a state and how it develops into the immediate future. E.g., a vessel's position or a water tank's temperature. Differential equations are the typical representation of the evolution of the state over time.

To better highlight how we utilize optimal control in our work, it can be helpful to classify them based on some properties. It does not exist a hybrid optimization problem that handles all types of problems well, and the classification of the dynamic systems helps formulate a solvable problem.

Suppose we consider a state space $\boldsymbol{X}$. A relevant classification of a dynamic system is whether the state space is continuous or discrete. First, we define the state as x. For a continuous state space, x can take values from the real numbers $\mathbb{R}$, like an ASV's position in a harbor area. If the same state space were discrete, one would sample the harbor area into a grid, and the state x could take values from the finite set of the grid.

We usually split dynamic systems into discrete-time systems and continuous-time systems. In a discrete-time system, a finite number of points separates any two points in time t_0 and t_1, while in a continuous-time system, an infinite number of points separates any two points in time t_0 and t_1. The most common example of a continuous-time system is physical time, and a sampling of physical time would form a discrete-time system.

Additionally, we can classify dynamic systems into time-invariant and time-variant systems. This classification describes whether the system's state evolution is dependent on the moment of observation. A dynamic system describing the earth's thermodynamic response to solar radiance will be a time-variant system because the earth's ozone layer will vary based on the moment of observation. However, most dynamic systems are not dependent on the moment of observation, making them time-invariant.

In this thesis, we will consider a docking scenario of an ASV, which is a continuous time-invariant nonlinear system.

2.4.2 Continuous optimal control problems

As we investigate a docking scenario of an ASV, we can define a continuous time-invariant nonlinear optimal control problem (OCP) without terminal constraints, as in Kirches et al. (2012),

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_{t_0}^{t_0+T} L(x(t),u(t))\,dt && \text{(2.4.1a)}\\
\text{subject to} \quad & x(t_0) = x_0, && \text{(2.4.1b)}\\
& \dot{x}(t) = f(x(t),u(t)), \quad t \in [t_0, t_0+T], && \text{(2.4.1c)}\\
& h(x(t),u(t)) \leq 0, \quad t \in [t_0, t_0+T]. && \text{(2.4.1d)}
\end{aligned}$$

Where x is the state variables, u is the control variable, L is the integral cost term, t_0 is the initial time, and T is the time horizon. The OCP also has to satisfy initial constraints (2.4.1b), state dynamics (2.4.1c), and path constraints (2.4.1d).

Numerical solutions of an OCP are usually divided into three groups, namely, state space, indirect, and direct methods. The state space method is mainly viable for small state dimension problems because of the curse of dimensionality (Horowitz et al. (2014)). The indirect and direct methods are more similar and broader in use. Indirect methods are often very accurate and outperform most other methods in terms of accuracy, even for high dimensional states. One of the major drawbacks is the need to formulate first-order necessary conditions for every problem instance (Böhme and Frank (2017)), and state constraints can be challenging to implement. Direct methods discretize the complete system and formulate the entire problem as finite-dimensional NLP. They tend to be less accurate than indirect methods but still benefit from being more flexible, numerically stable and handling all types of constraints well. Thus, making it more viable in a practical scenario.

2.4.3 Direct collocation method

To solve a continuous time-invariant nonlinear optimal control problem, we have opted for the direct collocation method, which is well known for exploiting the sparsity of the optimal control problem (OCP). The direct collocation method discretizes the state and control inputs onto a grid of discrete-time t_k where $k = 0, ..., N$ and N is the number of steps. Consequently, the OCP is transformed to a finite-dimensional nonlinear program.

First, we choose a set of $t_{k,i}$ collocation points on each collocation interval $[t_k, t_{k+1}]$, where $i = 0, ..., d$ and d is the number of collocation times. A polynomial $p_k(t, v_k) \in \mathbb{R}^n$ approximates the state trajectory x_k on the interval $[t_k, t_{k+1}]$ where the coefficients $v_k \in \mathbb{R}^{n_x(d+1)}$. From this, we get a set of collocation equations from Diehl and Gros (2017),

$$c_k(v_k, x_k, u_k) = \begin{bmatrix} v_{k,0} - s_k \\ \dot{p}_k(t_{k,1}, v_k) - f(v_{k,1}, t_{k,1}, u_k) \\ \vdots \\ \dot{p}_k(t_{k,d}, v_k) - f(v_{k,d}, t_{k,d}, u_k) \end{bmatrix} = 0, \tag{2.4.2}$$

where $v_{k,i} \in \mathbb{R}^{n_x}$, $i = 0, ..., d$, u_k is the discretized control and $f(v_{k,i}, t_{k,i}, u_k) = f(p_k(t_{k,i}, v_k), t_{k,i}, u_k)$ for the ODE $\dot{x} = f(x(t), t)$. In addition to solving the collocation equations, Diehl and Gros (2017) express that the interval boundaries require continuity, i.e., we require that

$$p_k(t_{k+1}, v_k) - u_{k+1} = 0, \tag{2.4.3}$$

holds for $k = 0, ..., N$.

Discretizing the integral from 2.4.1a leads to $\int_{t_k}^{t_{k+1}} L_k(x_k, u_k)\, dt$. A quadrature formula should approximate the discretized integral using the coefficients v_k, with the resulting approximation defined as $l_k(v_k, x_k, u_k)$.

We end up with a sparse NLP as described by Diehl and Gros (2017),

$$
\begin{aligned}
\min_{v,x,u} \quad & \sum_{k=0}^{N-1} l_k(v_k, x_k, u_k) && && \text{(2.4.4a)}\\
\text{subject to} \quad & x_0 - x(0) = 0, && && \text{(2.4.4b)}\\
& c_k(v_k, x_k, u_k) = 0, && k = 0, ..., N-1, && \text{(2.4.4c)}\\
& p_k(t_{k+1}, v_k) - x_{k+1} = 0, && k = 0, ..., N-1, && \text{(2.4.4d)}\\
& h(x_k, u_k) \leq 0, && k = 0, ..., N-1, && \text{(2.4.4e)}
\end{aligned}
$$

where 2.4.4a is the approximated integral term, 2.4.4b is the initial constraints, 2.4.4c is the collocation equations, 2.4.4d is the continuity equations, and 2.4.4e is the path constraints.

Chapter 3

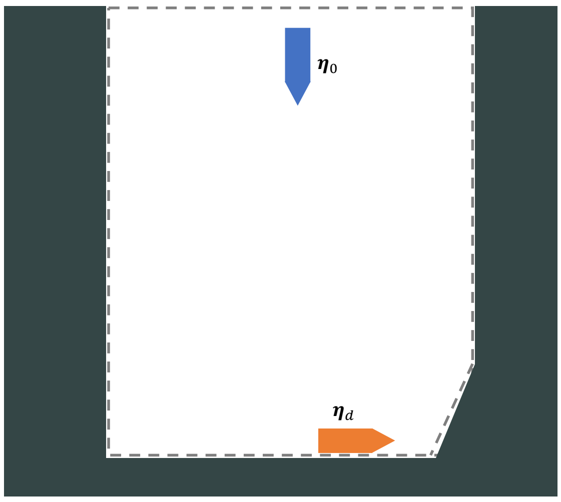
Problem description and methodology

The following chapter will present the problem description and implementation details on our two methodologies for automatic docking and dynamic and static obstacle avoidance of an ASV.

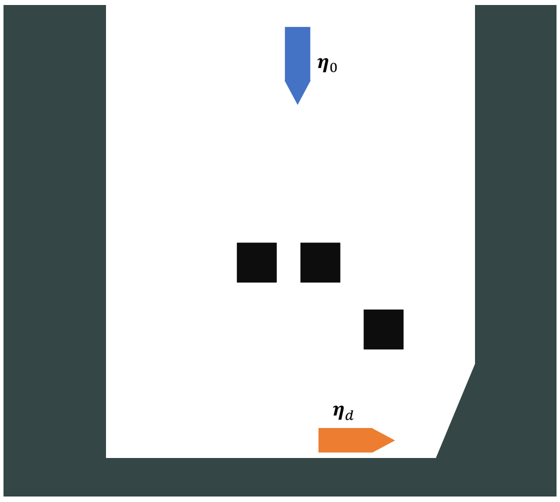
3.1 Problem formulation

This thesis considers a vessel with dynamics as described in Chapter 2. A convex polygonal space of a harbor area, created from map data, encloses the vessel's initial pose η_0 and dock pose η_d shown as the stapled line in Figure 1a. A variable amount of obstacles are then placed in different locations to obstruct the direct path to the dock pose, see Figure 1b. Our methodologies' goal is to find feasible obstacle-free docking trajectories in scenarios similar to Figure 1b where obstacles interfere with the direct path to the dock pose.

First and foremost it is necessary that our methodologies keep the vessel safe and avoid colliding with obstacles. Secondly, we want to avoid local optima and dock the vessel correct for all harbor configurations. These objectives leads to several different challenges related to non-linear optimization, path planning and low-level motion control.



(1a) Initial problem where a convex polygon encloses the initial pose η_0 and the dock pose η_d



(1b) Squared obstacles are placed to obstruct the vessel's direct path from the initial pose η_0 to the dock pose η_d

Figure 3.1.1: Initial problem where obstacles obstruct the direct path to the dock pose

3.2 Methodology

3.2.1 Overview

This thesis presents two multi-stage systems for automatic docking of an ASV. Figure 3.2.1 shows the general workflow of the methods.

Method A applies convex spatial constraints in the same manner as in Martinsen et al. (2019), Bitar et al. (2020), Martinsen et al. (2020), and Ødven (2021). We decompose the non-convex harbor area into multiple obstacle-free convex polygons. Thus, we can use the equivalent collision avoidance approach as in the work mentioned above. In short, we constrain the vessel to only operate in safe areas that are created based on the harbor and obstacle geometries. These areas are connected by overlapping areas so that the vessel can safely travel from the initial pose to the dock pose.

The framework of method A is that it applies computational geometry for polygon decomposition, a novel approach we refer to as *tangential decomposition*. From the decomposition, we create a graph of neighboring polygons. We then use the A* algorithm to find the shortest path to traverse in terms of connected polygons to reach the dock pose. The A* finds a set of waypoints, which are then used in an optimization problem to find a feasible collision-free docking trajectory.

The most significant difference between the two methods proposed is that method B does not create the same convex areas for the vessel to be safe when maneuvering. Instead, method B constrain the obstacles as unsafe regions and exclude them from the operational space. For method B, we apply constrained Delaunay triangulation for polygon decomposition, which allows us to find the medial axis of the obstacle-free space. We create waypoints with the A* algorithm and form a straight-lined initial guess to find a docking trajectory from an optimization problem.

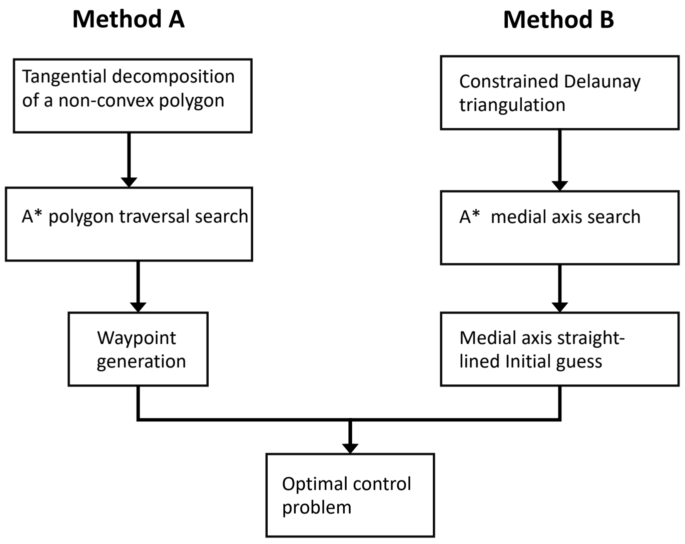

Method A
Method B
Tangential decomposition of a non-convex polygon
A* polygon traversal search
Waypoint generation
Constrained Delaunay triangulation
A* medial axis search
Medial axis straight-lined Initial guess
Optimal control problem

Figure 3.2.1: Block diagram showing the general workflow of method A and B

3.2.2 Model dynamics

In the simulations we present in the thesis, we consider two different vessels. A large tanker model, the Northern Clipper (Fossen et al. (1996)), and a small passenger ferry, the milliAmpere (Brekke et al. (2022)). Testing our methods with different scenarios, port geometry, and two different vessel models will help us understand the robustness and stability of our methodology in a variety of different conditions. Both vessels are described by the dynamics of a 3 DOF vessel presented in Section

2.2. For complete details on the models and the parameters used, see appendix A.

milliAmpere

The milliAmpere is an underactuated square-shaped research vessel at the Centre for Research-based Innovation (SFI) Autoship shown in Figure 3.2.2. For this vessel we will run closed-loop control simulations, and for that reason, we describe both a planning model and a simulation model of the vessel. We propose a 3 DOF surge-decoupled model for the simulations, described by Pedersen (2019), where the matrices $\boldsymbol{M}, \boldsymbol{C}(\boldsymbol{\nu})$, and $\boldsymbol{D}(\boldsymbol{\nu})$ are given as,

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix}, \tag{3.2.1}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & c_{13}(\boldsymbol{\nu}) \\ 0 & 0 & c_{23}(\boldsymbol{\nu}) \\ c_{31}(\boldsymbol{\nu}) & c_{32}(\boldsymbol{\nu}) & 0 \end{bmatrix}, \tag{3.2.2}$$

$$\boldsymbol{D}(\boldsymbol{\nu}) = \begin{bmatrix} d_{11}(\boldsymbol{\nu}) & 0 & 0 \\ 0 & d_{22}(\boldsymbol{\nu}) & d_{23}(\boldsymbol{\nu}) \\ 0 & d_{32}(\boldsymbol{\nu}) & d_{33}(\boldsymbol{\nu}) \end{bmatrix}. \tag{3.2.3}$$

The planning model is a simpler diagonalized model of the simulation model dynamics, described by Bitar et al. (2020),

$$\boldsymbol{S}\boldsymbol{M}_p\dot{\boldsymbol{\nu}}_p + \boldsymbol{C}_p(\boldsymbol{\nu}_p)\boldsymbol{\nu}_p + \boldsymbol{D}_p(\boldsymbol{\nu}_p)\boldsymbol{\nu}_p = \boldsymbol{\tau}_p, \tag{3.2.4}$$

where the matrices are given as,

$$\boldsymbol{S} = \begin{bmatrix} 2.5 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 5 \end{bmatrix}, \tag{3.2.5}$$

$$\boldsymbol{M}_p = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix}, \tag{3.2.6}$$

$$\boldsymbol{C}_p(\boldsymbol{\nu}_p) = \begin{bmatrix} 0 & 0 & -m_{22}v_p \\ 0 & 0 & m_{11}u_p \\ m_{22}v_p & -m_{11}u_p & 0 \end{bmatrix}, \tag{3.2.7}$$

$$\boldsymbol{D}_p(\boldsymbol{\nu}_p) = \begin{bmatrix} d_{11,p}(u_p) & 0 & 0 \\ 0 & d_{22,p}(v_p) & 0 \\ 0 & 0 & d_{33,p}(r_p) \end{bmatrix}. \tag{3.2.8}$$

For the thrust configuration of the MilliAmpere, we make some simplifications from (2.2.4) where we neglect the azimuth angle $\boldsymbol{\alpha}$. We want to include control allocation in the optimization problem, thus giving the thrust configuration as,

$$\boldsymbol{\tau} = \boldsymbol{T}\boldsymbol{f}, \tag{3.2.9}$$

where the thrust configuration for two thrusters on the center line of the vessel is,

$$\boldsymbol{T}\boldsymbol{f} = \begin{bmatrix} f_{x,1} & f_{x,2} \\ f_{y,1} & f_{y,2} \\ f_{y,1}\,L_{x,1} & f_{y,2}\,L_{x,2} \end{bmatrix}, \tag{3.2.10}$$

$f_{i,j}$ is the force of thruster j in direction i, and $L_{i,j}$ is the position of thruster j with respect to the body-fixed CO.

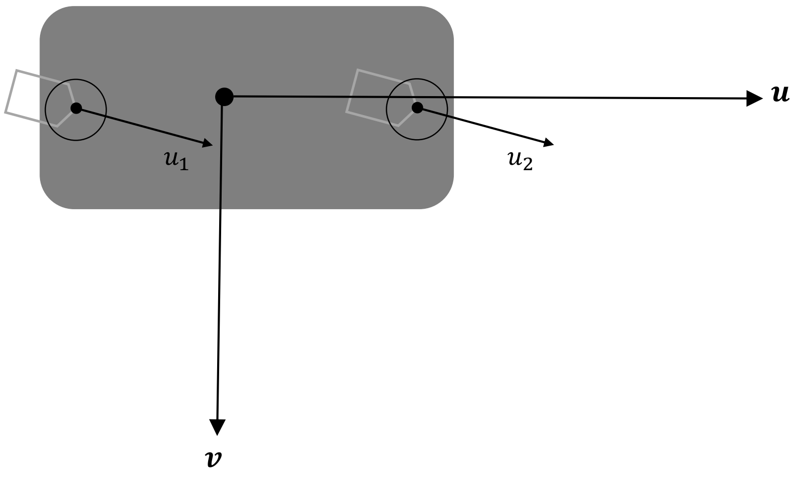


Figure 3.2.2: Vessel model and thrust configuration of milliAmpere. One azimuth thruster in the front, and one in the aft of the vessel

Northern Clipper

The Northern Clipper is a fully actuated vessel with a more traditional marine hull shown in Figure 3.2.3. We describe the vessel by a 3 DOF maneuvering model where a simulation of a DP control mode in Fossen et al. (1996) lead to the dynamics,

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}. \tag{3.2.11}$$

The inertia matrix $\boldsymbol{M} = m\boldsymbol{N}\boldsymbol{M}_{bis}$, and dampening matrix $\boldsymbol{D}(\boldsymbol{\nu}) = m\sqrt{\frac{g}{L}}\boldsymbol{N}\boldsymbol{D}_{bis}\boldsymbol{N}$. The constants m, g, L are the mass, gravitational acceleration, and length of the ship. The matrices are given by

$$\boldsymbol{M}_{bis} = \begin{bmatrix} 1 - X_{\dot{u}} & 0 & 0 \\ 0 & 1 - Y_{\dot{v}} & x_G - Y_{\dot{r}} \\ 0 & x_G - Y_{\dot{r}} & k_z^2 - N_{\dot{r}} \end{bmatrix}, \tag{3.2.12}$$

$$\boldsymbol{D}_{bis} = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & x_G - Y_r \\ 0 & -N_v & -N_r \end{bmatrix}, \tag{3.2.13}$$

$$\boldsymbol{N} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & L \end{bmatrix}. \tag{3.2.14}$$

The thrust configuration is given as

$$\boldsymbol{\tau} \; = \; \boldsymbol{T}(\boldsymbol{\alpha})\,\boldsymbol{F}, \tag{3.2.15}$$

with two azimuth thrusters in the stern and one tunnel thruster in the bow. The thrust configuration matrix is

$$\boldsymbol{T}(\boldsymbol{\alpha}) \; = \; \begin{bmatrix} \cos(\alpha_1) & \cos(\alpha_2) & 0 \\ \sin(\alpha_1) & \sin(\alpha_2) & 1 \\ l_{x,1}\,\sin(\alpha_1) \, - \, l_{y,1}\,\cos(\alpha_1) & l_{x,2}\,\sin(\alpha_2) \, - \, l_{y,2}\,\cos(\alpha_2) & l_{x,3} \end{bmatrix}, \tag{3.2.16}$$

where α_j is the angle of thruster j, and $L_{i,j}$ is the position of thruster j with respect to the body-fixed CO.

The thrusters also have some physical constraints that we consider in the optimization problem. Specifically, it has force saturation (3.2.17a), azimuth angle saturation (3.2.17b), and angular velocity saturation (3.2.17c) (Martinsen et al. (2019)).

$$f_{i,min} \leq f_i \leq f_{i,max}, \tag{3.2.17a}$$

$$\alpha_{i,min} \leq \alpha_i \leq \alpha_{i,max}, \tag{3.2.17b}$$

$$\dot{\alpha}_{i,min} \leq \dot{\alpha}_i \leq \dot{\alpha}_{i,max}. \tag{3.2.17c}$$

The azimuth thrusters have feasible angles from $[-80, 260], [-260, 80]$ degrees. Figure 3.2.3 shows illegal thruster angles as the red areas. The azimuth thrusters also have a maximum angular velocity of $|\dot{\alpha}| \leq 2\pi/30$. The thrust force saturation for the azimuth thrusters is $m/30$, and for the tunnel thruster, it is $\pm m/60$. Figure 3.2.3 shows the entire thrust configuration.

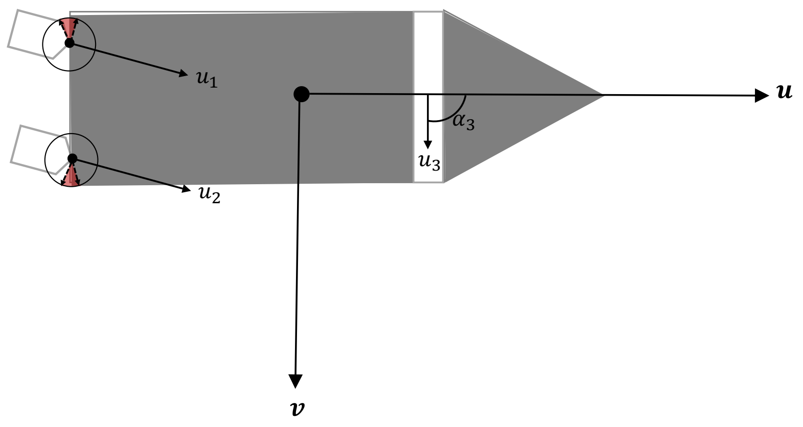


Figure 3.2.3: The Northern clipper vessel model and its thrust configuration with two azimuth thrusters in the aft of the vessel, and one tunnel thruster in the front. The red regions mark illegal thruster orientation

3.2.3 Method A

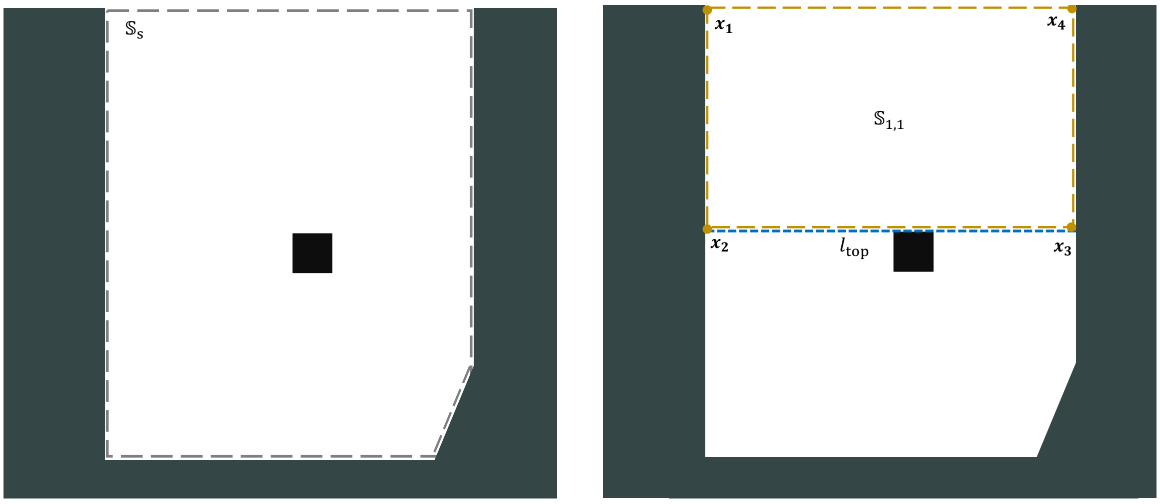
Tangential decomposition of a non-convex polygon

When introducing obstacles in the harbor area, the convex spatial constraints (Martinsen et al. (2019), Bitar et al. (2020), Martinsen et al. (2020)) are more intricate and complex to apply as safety guarantee for the vessel.

To take advantage of such convex spatial constraints, we need to decompose the harbor area into convex polygons of free space. Let us consider a harbor geometry with the initial convex polygon $\mathbb{S}_s$ and an obstacle modeled as a square, as shown in Figure 4a. To decompose the polygon $\mathbb{S}_s$, we extend the obstacle's edges with a line l_i where $i = [top, bottom, right, left]$. The line l_i will intersect the set $\mathbb{S}_s$ and allows us to create new obstacle-free convex sets $\mathbb{S}_{k,s}$ around an obstacle $\boldsymbol{O}_k$ where $k = 1, ..., n_k$, and n_k is the number of obstacles. In the simple case of one obstacle, we know, for instance, that the set $\mathbb{S}_{1,1}$ will be obstacle-free as long as,

$$l_{top} \geq \boldsymbol{x}_j \in \mathbb{S}_s, \tag{3.2.18}$$

for every point $\boldsymbol{x}_j$ where $j = 1, ..., n$ and n is the number of points in the set $\mathbb{S}_s$. We show the resulting decomposed set $\boldsymbol{S}_{1,1}$ in Figure 4b. Using the property of (3.2.18), we can create all the new regions $\mathbb{S}_{k,s}$ similarly.



(4a) The harbor geometry with the convex set $\mathbb{S}_s$ enclosing an obstacle

(4b) The creation of the convex set $\mathbb{S}_{1,1}$ from the line l_{top} and set $\mathbb{S}_s$

Figure 3.2.4: A decomposition from an initial harbor geometry and obstacle position

As the ship hulls are modeled as convex polygons and not point masses, we intentionally decompose so that the convex sets $\mathbb{S}_{k,s}$ creates overlapping regions $\mathbb{O}_{s_1,s_2}$, shown in Figure 3.2.5. We take advantage of this property when generating waypoints to compute a safe and feasible trajectory.

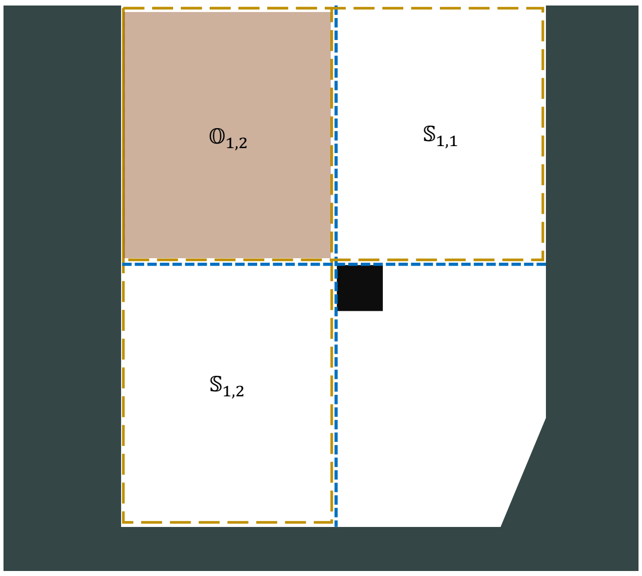


Figure 3.2.5: The overlapping region $\mathbb{O}_{1,2}$ created by the convex sets $\mathbb{S}_{1,2}$ and $\mathbb{S}_{1,1}$

The decomposition when dealing with a single obstacle is similar to what we presented in the project report (Ødven (2021)). For multi-obstacle scenarios, the decomposition becomes more complex. Thus, we have developed an extension of the proposed approach from Ødven (2021). Let us consider the case of Figure 3.2.6 and the decomposition around obstacle $\boldsymbol{O}_3$ and line l_{top}. Using (3.2.18), we can not guarantee an obstacle-free decomposition in such cases.

To define the multi-obstacle decomposition, let us first define the set $\mathbb{D}$ as the center $\boldsymbol{c}_i = (c_{i,x}, c_{i,y})$ of the obstacles and n_o as the number of obstacles where $c_{i,y} < c_{3,y}$. We can then define the multi-obstacle decomposition using the pseudocode from Algorithm 2. The algorithm computes a maximum of n_o+1 different decompositions corresponding to the tangent l_{top}, as shown in Figure 3.2.6.

We have designed the decomposition to find the polygons with the largest areas constrained by the opposite-side tangent of the remaining obstacles. In the case of Figure 3.2.6, for instance, the opposite-side tangent is the bottom tangent.

The pseudocode of Algorithm 2 is for a specific tangent l_{top} to provide a more straightforward explanation. However, the same approach applies to every tangent of the obstacle.

Algorithm 2: Tangential decomposition (l_{top})

```
R ← InitializeEmptyList();
O ← SortObstacles(Key = $c_y \in \mathbb{D}$);
for $i = 0$ to $n_o$ do
    $l_{bot}$ ← ObstacleTangent($O_i$, bottom);
    $\boldsymbol{P} \leftarrow \{\boldsymbol{x}_j \mid l_{bot} \leq \boldsymbol{x}_j \leq l_{top}, \forall \boldsymbol{x}_j \in \mathbb{S}\}$;
    while $i > 0$ do
        $l_{left}$ ← ObstacleTangent($O_i$, left);
        $l_{right}$ ← ObstacleTangent($O_i$, right);
        $\boldsymbol{P} \leftarrow \{\boldsymbol{x}_j \mid l_{left} \leq \boldsymbol{x}_j \leq l_{right}, \forall \boldsymbol{x}_j \in \mathbb{P}\}$;
        $i \leftarrow i - 1$;
    R ← Append($\boldsymbol{P}$);
return R;
```

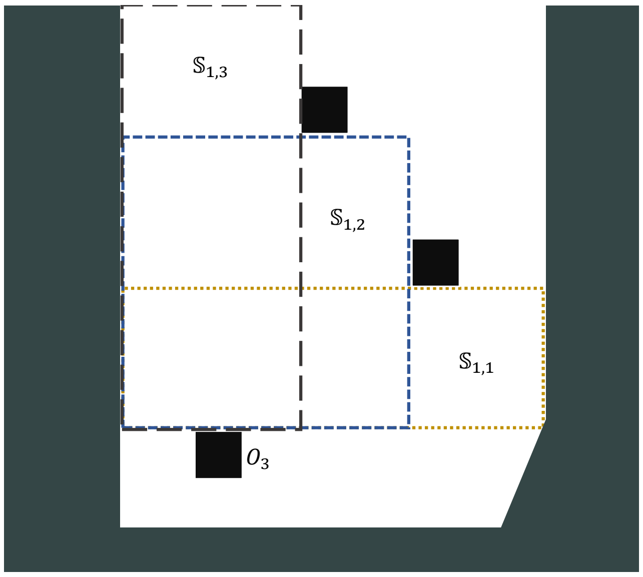


Figure 3.2.6: The three decompositions $\mathbb{S}_{1,1}$, $\mathbb{S}_{1,1}$, and $\mathbb{S}_{1,1}$ for the line l_{top} and obstacle $\boldsymbol{O}_3$

Graph and waypoint generation

To find the shortest path to the dock pose, we create a graph of the convex polygons $\mathbb{S}_{k,s}$ where the edges connecting them are the overlapping polygons $\mathbb{O}_{i,j}$, shown in Figure 3.2.7. The A* algorithm then finds the shortest sequence of convex polygons to traverse. The sequence is from the convex polygon $\mathbb{S}_{k,s}$, where the initial pose $\boldsymbol{\eta}_0 \subseteq \mathbb{S}_{k,s}$, to the convex polygon $\mathbb{S}_{k,d}$ where the dock pose $\boldsymbol{\eta}_d \subseteq \mathbb{S}_{k,d}$.

With waypoints, we can partition the optimization problem into less complex problems where we optimize with respect to waypoints and constrain the optimization to connected convex polygonal spaces. We define the waypoints $\boldsymbol{\eta}_{w,j} = [x_{w,j}, y_{w,j}, \psi_{w,j}]^\top$ as the centroid (x, y) of the overlapping polygons $\mathbb{O}_{i,j}$ with a constant heading angle $\psi_{w,j}$ in the direction of travel. Where $j = 1, ..., n_w$ and n_w is the number of nodes in the path found by the A*. Figure 3.2.8 shows how each waypoint has a corresponding obstacle-free convex polygon enclosing it. The figure also shows how the overlapping regions act as a safety zone for safely switching to the next waypoint. To entice the algorithm to choose paths with larger overlapping areas, we implement weights on the overlapping areas $\mathbb{O}_{i,j}$ as follows,

$$1 - \left| \frac{3\hat{A}(\mathbb{S}_b)^2}{\hat{A}(\mathbb{O}_{i,j}) + \epsilon} \right|, \tag{3.2.19}$$

where $\hat{A}(x)$ describes the normalized area of a set x with respect to the largest overlapping area, $\max(A(\mathbb{O}_{i,j}))$. The small constant $\epsilon > 0$ is to avoid division by zero. This weight distribution allows negative weight. However, negative weight implies that the overlapping area is not sufficiently large compared to the area of the vessel hull, and the overlapping area will therefore be removed as an edge in the graph. By creating waypoints in such a way, we can ensure the vessel's safety while helping avoid local minima in the path to the dock pose.

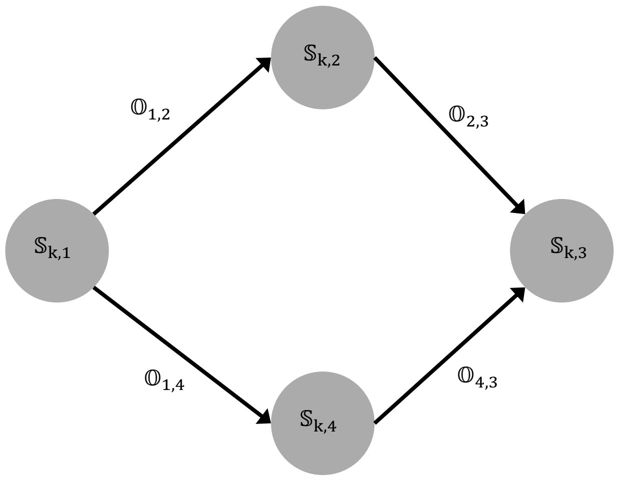


Figure 3.2.7: A graph corresponding to the scenario in Figure 4a, where $\boldsymbol{\eta}_0 \subseteq \mathbb{S}_{k,1}$ and $\boldsymbol{\eta}_d \subseteq \mathbb{S}_{k,3}$

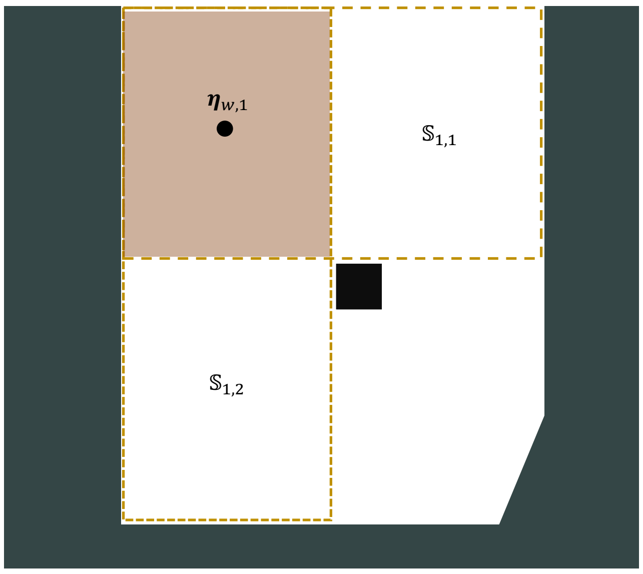


Figure 3.2.8: Waypoint defined as the centroid of the overlapping region $\mathbb{O}_{1,2}$

3.2.4 Method B

Constrained Delaunay triangulation and medial axis of obstacle-free space

To overcome the problems of a non-convex harbor area we apply constrained Delaunay triangulation for decomposition. A potential constrained Delaunay triangulation of the case in Figure 3.2.4 is shown in Figure 3.2.9. From the CDT, we get the medial axis from the lines connecting the centroids of the triangles, illustrated in Figure 3.2.9. Using the medial axis, we create a set of waypoints or straight-line segments to plan the path to the dock pose. Using the medial axis is to ensure that the line segments are equally far from the obstacles as the quay, as this property makes the medial axis advantageous when planning paths in small spaces, e.g., when docking in a tight harbor. However, we do not get an optimal medial axis when dealing with polygonal holes, as shown in Figure 3.2.9.

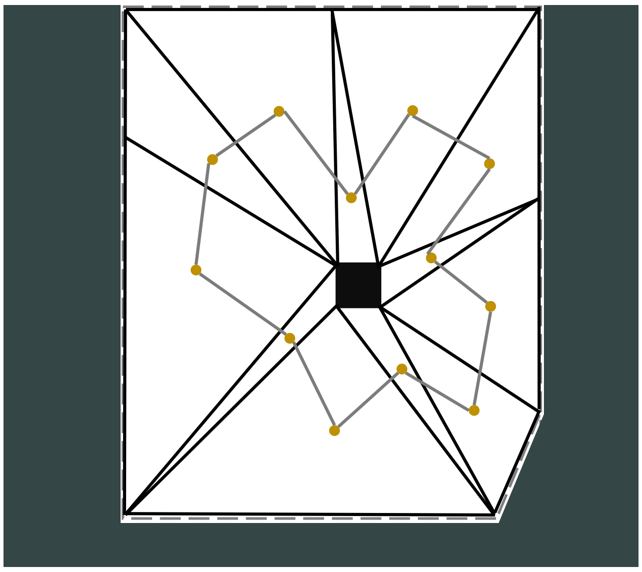


Figure 3.2.9: The CDT and medial axis of a convex polygon enclosing one obstacle

Straight-lined initial guess

Figure 3.2.10 shows the step-by-step process of how we go from a constrained Delaunay triangulation to an initial guess used in an optimization problem. By connecting the initial and dock pose to their closest point on the medial axis, we create line segments connecting the initial pose to the dock pose, as shown in Figure 10a. The next step is to find the shortest path to the dock pose. We use the A* algorithm on the graph to find the path shown in Figure 10b. As we base the path on equidistance, it can create a sub-optimal path in terms of motion control of a marine vessel. It is generally unwanted to have many short-distance waypoints with swift turning. Because of that, we apply a waypoint-reduction algorithm from Bitar et al. (2019) and end up with the initial guess shown in Figure 10c. We are not taking the vessel's kinematics and dynamics into account for the initial guess and assuming constant velocity. Thus, the initial guess is uniformly distributed over our optimization horizon. The waypoint-reduction algorithm reduces the initial path to a path with as few waypoints and straight-line segments as possible, and the pseudocode is presented in Algorithm 3.

Algorithm 3: Waypoint-reduction algorithm

```
Procedure REDUCE
$i \leftarrow N_*$;
$\mathbb{P} \leftarrow$ InitializePath($\boldsymbol{p}_i^*$); while $i > 1$ do
    for $j = 1$ to $i - 1$ do
        if $\neg$ Collision($\boldsymbol{p}_i^*$,$\boldsymbol{p}_j^*$) then
            AddPoint($\mathbb{P}$,$\boldsymbol{p}_j^*$);
            $i \leftarrow j$;
            break;
```

Ellipse constraint

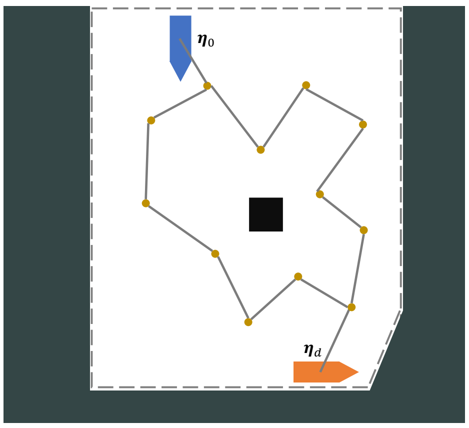
The medial axis does not consider the ship's hull, which means that if the vessel were to follow a path generated by the medial axis, it could collide with obstacles. As a safety guarantee, we model obstacles as elliptic constraints in the optimization problem to exclude them from the operational space. First, we describe an ellipse as,

$$\frac{x - x_c}{x_a}^2 + \frac{y - y_c}{y_a}^2 \geq 1, \tag{3.2.20}$$

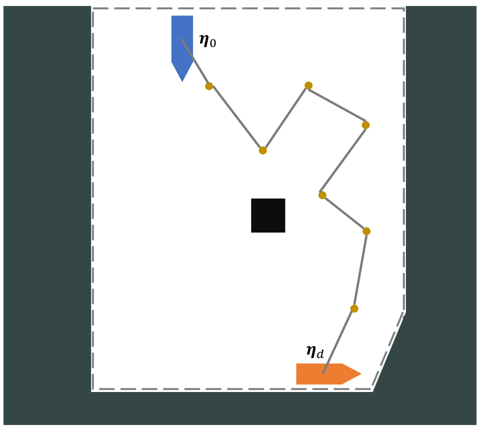
where x_c, y_c is the center of the ellipse in the x and y axes, and the elliptic axes is x_a and y_a, respectively. From (3.2.20) we can describe a set of inequalities for a position $\boldsymbol{\eta}_p = [x, y]^\top$ similar to Bitar et al. (2019),

$$-\log\left[\left(\frac{x - o_{x,i}}{x_{a,i}}\right)^2 + \left(\frac{y - o_{y,i}}{y_{a,i}}\right)^2 + \epsilon\right] + \log(1 + \epsilon) \leq 0, \tag{3.2.21}$$

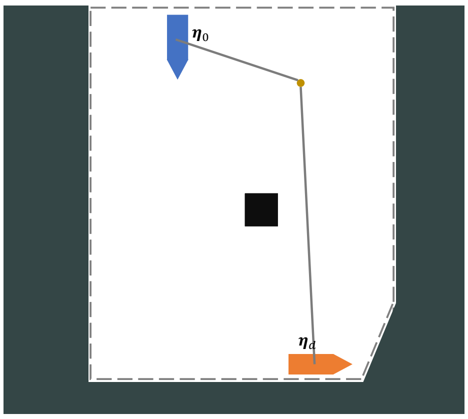
where $o_{x,i}$ and $o_{y,i}$ is the obstacle center in Cartesian coordinates (x, y) of obstacle i, $i = 1, .., n$, and n is the number of obstacles. To avoid problems with singularity when $x \rightarrow o_{x,i}$ and $y \rightarrow o_{y,i}$ we add a small constant $\epsilon > 0$.



(10a) Connecting the initial η_0 and dock pose η_d to the medial axis



(10b) The A* finds points the optimal path to traverse



(10c) The initial guess used in the optimization after applying the waypoint reducing algorithm

Figure 3.2.10: Step-by-step from the medial axis to the initial guess used in the optimization

3.2.5 Convex spatial constraint

In order to avoid collisions, both methods utilize that a convex polygon can be described as linear inequalities and use these as inequality constraints in an optimization problem. First, we define a convex polygon $\mathbb{S}_s$ enclosing the harbor and describe $\mathbb{S}_s$ as a solution to the linear inequalities,

$$\boldsymbol{A}\boldsymbol{v}_i \leq \boldsymbol{b}, \quad \text{where } i = 1, ..., s, \tag{3.2.22}$$

$\boldsymbol{A} \in \mathbb{R}^{s\times 3}$, $\boldsymbol{b} \in \mathbb{R}^{s\times 1}$, $\boldsymbol{v}_i \in \mathbb{R}^2$ is vertex i's Cartesian position (x, y), and s is the number of vertices.

To take advantage of (3.2.22) for collision avoidance, we define the set $\mathbb{S}_b$ as the convex set describing the vessel's hull with each vertex as $\boldsymbol{x}_i^b \in \text{Vertex}(\mathbb{S}_b)$. shown in Figure 3.2.11. By using (3.2.22) while taking into account the vessel's hull, we can describe a set of linear inequalities to include in an optimization problem as,

$$\boldsymbol{A}\left(\boldsymbol{R}_1(\psi)\boldsymbol{x}_i^b + \begin{bmatrix} x \\ y \end{bmatrix}\right) \leq \boldsymbol{b} \quad \forall\, \boldsymbol{x}_i^b \in \text{Vertex}(\mathbb{S}_b), \tag{3.2.23}$$

where $\boldsymbol{R}_1$ is the rotation from the body frame to NED,

$$\boldsymbol{R}_1(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}. \tag{3.2.24}$$

The matrices $\boldsymbol{A} \in \mathbb{R}^{s\times 3}$ and $\boldsymbol{b} \in \mathbb{R}^{s\times 1}$. The vertex $\boldsymbol{x}_i^b \in \mathbb{R}^2$ is vertex i's Cartesian position (x, y) and $[x, y]^\top$ is the vessel CO with respect to the North-East-Down (NED) reference frame. From (3.2.23) we can constrain the vessel to only operate inside a desired region, as shown in Figure 3.2.11.

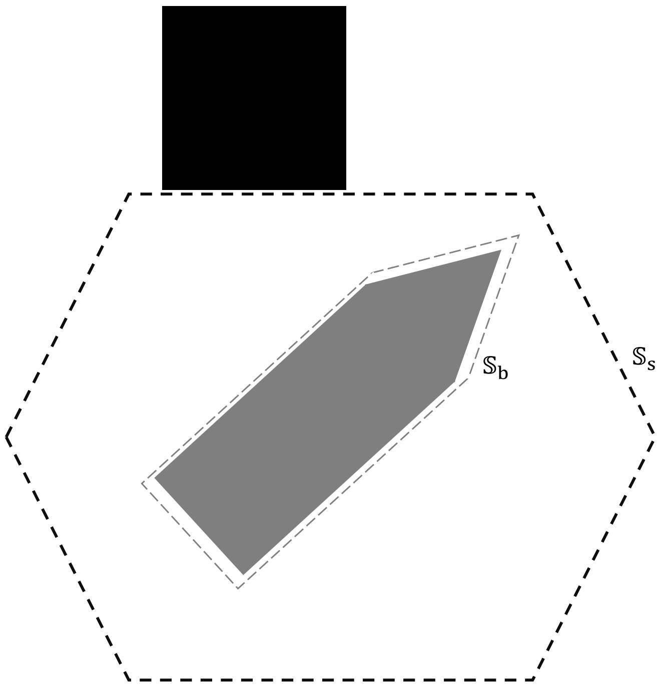


Figure 3.2.11: The vessel hull $\mathbb{S}_b$ constrained to operate inside the convex set $\mathbb{S}_s$

Method A

We use (3.2.23) somewhat differently for our two methods. For method A, we implement the same core idea as in the work of Martinsen et al. (2019): using the linear inequalities as a safety guarantee for the vessel and ensuring that the set $\mathbb{S}_s$ does not include any obstacles or land.

As explained in section 3.2.3, we partition the optimization problem and constrain it over smaller polygonal spaces. This approach leads to an extended set of the equations from (3.2.23),

$$\boldsymbol{A}_{s,j}\left(\boldsymbol{R}_1(\psi)\boldsymbol{x}_i^b + \begin{bmatrix} x \\ y \end{bmatrix}\right) \leq \boldsymbol{b}_{s,j} \quad \forall\, \boldsymbol{x}_i^b \in \text{Vertex}(\mathbb{S}_b), \tag{3.2.25}$$

where we have a set of linear inequalities for each convex polygon $\mathbb{S}_{k,s}$ we traverse, where $j = 0, ..., n_w$. The inequalities (3.2.25) guarantee the vessel's safety for method A and ensure we maneuver in obstacle-free space.

Method B

Figure 3.2.12 shows how a convex spatial constraint and elliptic constraints constrain the vessel to operate in obstacle-free space. We model the obstacles as elliptic constraints in the optimization problem for method B, explained in section 3.2.4. This approach allows us to use a more straightforward and less computationally expensive method when creating the set $\mathbb{S}_s$. We define $\mathbb{S}_s$ as the largest convex set not including any land. Hence, the convex polygon $\mathbb{S}_s$ only guarantees that we do not collide with the quay, which means we can directly implement the convex spatial constraint from (3.2.23).

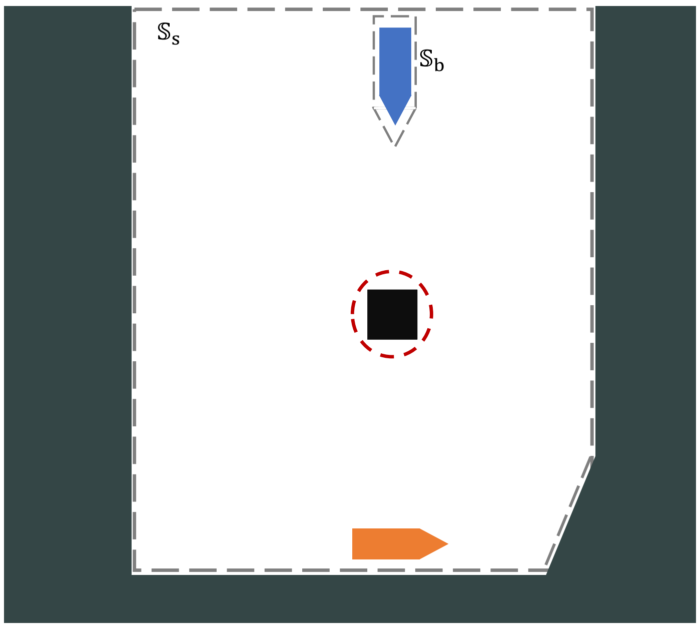


Figure 3.2.12: Convex spatial constraint $\mathbb{S}_s$ and elliptic constraint marked in red constrain the vessel $\mathbb{S}_b$ to operate in obstacle-free space

3.2.6 Optimal control problem

Both methods use optimization as open-loop control and as a planner for a trajectory tracking DP-controller. The general formulation of the problem is,

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{s}} \quad & \int_{t_o}^{t_0+T} \left\{ l_{m,v}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{s}(t)) \right\} dt, && \text{(3.2.26a)}\\
\text{subject to} \quad & \dot{\boldsymbol{x}} = f(\boldsymbol{x}(t),\boldsymbol{u}(t)), && \text{(3.2.26b)}\\
& \boldsymbol{x}_{min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{max}, && \text{(3.2.26c)}\\
& \boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max}, && \text{(3.2.26d)}\\
& h(\boldsymbol{x}(t),\boldsymbol{u}(t)) - \boldsymbol{s}(t) \leq \boldsymbol{0}, && \text{(3.2.26e)}\\
& \boldsymbol{s}(t) \geq \boldsymbol{0}, && \text{(3.2.26f)}\\
& \boldsymbol{x}(t_0) = \boldsymbol{x}_0, && \text{(3.2.26g)}
\end{aligned}
$$

where $l_{m,v}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{s}(t))$ is the objective term. The subscripts $m = [A, B]$ and $v = [MAM, NC]$ denote what method and vessel model we use. The variables $\boldsymbol{x}$, $\boldsymbol{u}$, and $\boldsymbol{s}$ are the vessel states, control inputs, and slack variables, respectively. The optimization problem has to be solved subject to the dynamic (3.2.26b) and physical constraints (3.2.26c, 3.2.26d) of the vessel models. Additionally, the optimization must satisfy path(3.2.26e), slack (3.2.26f), and initial constraints (3.2.26g).

milliAmpere

For the milliAmpere model (MAM), we use the objective function developed by Martinsen et al. (2020). They use a Pseudo-Huber function to have linear penalization of the positional error when far away from the dock pose. When closer to the dock pose, they use a quadratic penalty. According to Gros and Diehl (2013) and Gros and Zanon (2017), the Pseudo-Huber function improves numerical stability and has shown better performance for larger positional errors. For our positional penalization, the Pseudo-Huber function is as follows,

$$c_{x,y}(\boldsymbol{\eta}) = \delta^2\left(\sqrt{1+\frac{(x-x_d)^2+(y-y_d)^2}{\delta^2}}-1\right). \tag{3.2.27}$$

For the heading, we use a penalty function designed to choose an efficient heading when the vessel is far from the dock pose and gradually rotate it towards the dock pose as it approaches (Martinsen et al. (2020)). The heading penalty is,

$$c_\psi(\boldsymbol{\eta}) = \frac{1-\cos(\psi-\psi_d)}{2}e^{-\frac{(x-x_d)^2+(y-y_d)^2}{2\delta^2}}, \tag{3.2.28}$$

and the objective term $l_{m,MAM}$ is then,

$$l_{m,MAM} = c_{x,y}(\boldsymbol{\eta}) + 20c_\psi(\boldsymbol{\eta}) + \boldsymbol{\nu}^\top\boldsymbol{Q}\boldsymbol{\nu} + \boldsymbol{u}^\top\boldsymbol{R}\boldsymbol{u} + \boldsymbol{k}_s^\top\boldsymbol{s}, \tag{3.2.29}$$

where $\boldsymbol{Q} \in \mathbb{R}^{3\times3}$, $\boldsymbol{R} \in \mathbb{R}^{4\times4}$, and $\boldsymbol{k}_s \in \mathbb{R}^5$ are the weight matrices for the velocity $\boldsymbol{\nu}$, force $\boldsymbol{f}$, and slack variables $\boldsymbol{s}$, respectively.

Northern Clipper

For the Northern Clipper (NC) model, we use the objective term developed by Martinsen et al. (2019). The objective term is a simple quadratic penalty function where we penalize positional error, velocity, and force of the thrusters,

$$c_\eta = \|\boldsymbol{\eta}-\boldsymbol{\eta}_d\|^2_{\boldsymbol{Q}_\eta} + \|\boldsymbol{\nu}\|^2_{\boldsymbol{Q}_\nu} + \|\boldsymbol{f}\|^2_{\boldsymbol{R}_f} + \boldsymbol{k}_c^\top\boldsymbol{s}, \tag{3.2.30}$$

where the matrices $\boldsymbol{Q}_{\boldsymbol{\eta}} \in \mathbb{R}^{3\times3}$, $\boldsymbol{Q}_{\boldsymbol{\nu}} \in \mathbb{R}^{3\times3}$, $\boldsymbol{R}_{\boldsymbol{f}} \in \mathbb{R}^{3\times3}$, and $\boldsymbol{k}_c \in \mathbb{R}^3$ are the weights for the positional error $\boldsymbol{\eta} - \boldsymbol{\eta}_d$, velocity $\boldsymbol{\nu}$, force $\boldsymbol{f}$, and slack variables $\boldsymbol{s}$, respectively. The Northern Clipper vessel has some physical constraints. Hence, the objective term also incorporates a penalty on the rank deficiency to avoid singular thrust configurations (Martinsen et al. (2019)),

$$c_f = \frac{\rho}{\epsilon + \det\left(\boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{W}^{-1}\boldsymbol{T}^\top(\boldsymbol{\alpha})\right)}, \tag{3.2.31}$$

where ρ is the maneuverability constant, $\boldsymbol{T}$ is the thrust configuration matrix, $\boldsymbol{W}$ is the diagonal thrust weight matrix, and $\epsilon > 0$ is a small constant to avoid singularities. The objective term $l_{m,NC}$ for the Northern Clipper model is then as follows,

$$l_{m,NC} = c_\eta + c_f. \tag{3.2.32}$$

Method A

We partition the docking procedure into n_w optimization problems for method A. Therefore, we change the positional error $\boldsymbol{\eta} - \boldsymbol{\eta}_d$ in (3.2.27), (3.2.28), and (3.2.30) to $\boldsymbol{\eta} - \boldsymbol{\eta}_{w,j}$, where $j = 1, .., n_w$, and solve each problem over a horizon T. At time $t = T$, we switch to the subsequent problem and solve a new optimization problem with the vessel's current state as initial conditions and the desired pose as $\boldsymbol{\eta}_{w,j+1}$. The problem formulates as an NLP,

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{s}} \quad \int_{t_0}^{t_0+T} \left\{ l_{A,v}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{s}(t)) \right\} dt, \tag{3.2.33a}$$

$$\text{subject to} \quad \dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{3.2.33b}$$

$$\boldsymbol{x}_{min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{max}, \tag{3.2.33c}$$

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max}, \tag{3.2.33d}$$

$$\boldsymbol{A}_{s,j}\left(\boldsymbol{R}_1(\psi)\boldsymbol{v}_i^b + \begin{bmatrix} x \\ y \end{bmatrix}\right) \leq \boldsymbol{b}_{s,j}, \qquad \forall j,\ \forall \boldsymbol{v}_i^b \in \text{Vertex}(\mathbb{S}_b), \tag{3.2.33e}$$

$$\boldsymbol{s}(t) \geq \boldsymbol{0}, \tag{3.2.33f}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{3.2.33g}$$

where we minimize the objective term (3.2.33a) over the horizon $[t_0, t_0 + T]$. Subject to the dynamic state constraints (3.2.33b), state constraints (3.2.33c) and (3.2.33d), spatial constraints (3.2.33e), slack constraints (3.2.33f), and the initial conditions (3.2.33g).

Method B

As a result of using an initial guess as a warm-start, we do not partition the optimization problem. We optimize the positional error from the current pose to the dock pose $\boldsymbol{\eta} - \boldsymbol{\eta}_d$, where the desired pose $\boldsymbol{\eta}_d$ is the dock pose. From this, we can formulate the problem as below,

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{s}} \quad \int_{t_0}^{t_0+T} \Big\{ l_{B,v}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{s}(t)) \Big\} dt, \tag{3.2.34a}$$

$$\text{subject to} \quad \dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{3.2.34b}$$

$$\boldsymbol{x}_{min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{max}, \tag{3.2.34c}$$

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max}, \tag{3.2.34d}$$

$$\boldsymbol{A}_s \left(\boldsymbol{R}_1(\psi)\boldsymbol{v}_i^b + \begin{bmatrix} x \\ y \end{bmatrix} \right) \leq \boldsymbol{b}_s, \qquad \forall \boldsymbol{v}_i^b \in \text{Vertex}(\mathbb{S}_b) \tag{3.2.34e}$$

$$-\log\left[\left(\frac{v_0 - o_{x,i}}{x_{a,i}} \right)^2 + \left(\frac{v_1 - o_{y,i}}{y_< a, i} \right)^2 + \epsilon \right] + \log(1+\epsilon) \leq 0, \quad \forall \boldsymbol{v}_i^b \in \text{Vertex}(\mathbb{S}_b), \tag{3.2.34f}$$

$$\boldsymbol{s}(t) \geq \boldsymbol{0}, \tag{3.2.34g}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0. \tag{3.2.34h}$$

The spatial constraints (3.2.34e) are directly implemented as in (3.2.23), and the addition of elliptic constraints (3.2.34f) are the changes from method A. Apart from that, we have dynamic constraints (3.2.34b), state constraints (3.2.34c) and (3.2.34d), slack constraints (3.2.34g), and the initial conditions (3.2.34h) equivalent to method A.

3.2.7 Closed-loop control

The milliampere vessel use the separated planner and low-level DP controller from Bitar et al. (2020), as shown in Figure 3.2.13. Using a different planner and simulation model allows us to simulate a more realistic scenario where our controller has to account for modeling errors in our planning stage. Further details on why a separation-based planner and controller is beneficial for control purposes is mentioned in Bitar et al. (2020).

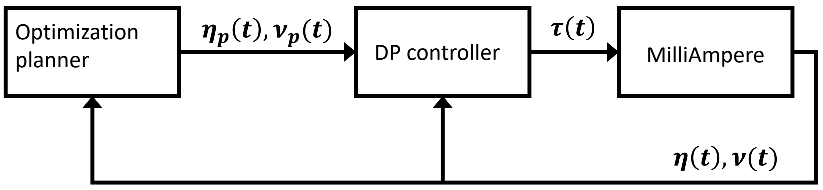


Figure 3.2.13: Block diagram of the separated optimization planner and low-level DP controller

Waypoint switching

We can improve the dock time for method A by applying a waypoint switching mechanism for closed-loop control. When running open-loop control, we have to perform all the control steps of the complete docking maneuver before a switch can occur. However, for closed-loop control, we can apply a switching mechanism. The switch to the next waypoint occurs whenever we are inside the overlapping region where our current waypoint is located, as shown in Figure 3.2.14. By switching to the next waypoint when we have just arrived at the overlapping region, we interfere with the planned path and bypass most of the docking procedure the vessel enters when closing in on a waypoint. In addition to saving time, a faster switching mechanism smooths the trajectory and gives a more human-like behavior.

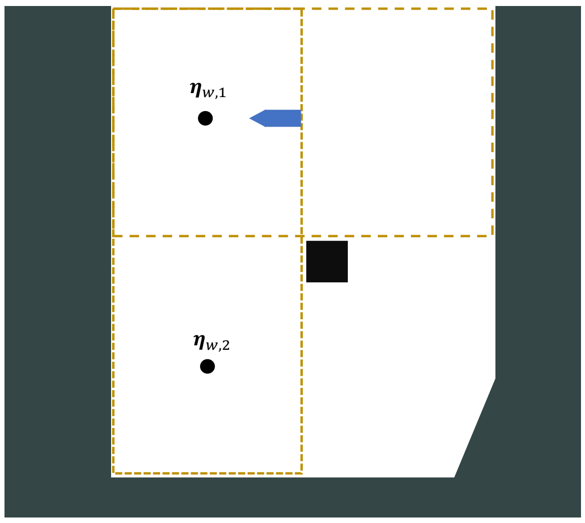


Figure 3.2.14: The instance where the vessel switches from tracking the waypoint $\eta_{w,1}$ to tracking the waypoint $\eta_{w,2}$

Chapter 4

Results and discussion

This chapter presents and discusses the simulation results from various simulation scenarios. We show a collection of diversified scenarios to showcase the two methods' strengths and weaknesses. The scenarios consist of open-loop control, separated optimization planner with closed-loop DP control, and dynamic obstacle avoidance. We show the methods' ability to handle vastly different vessel models. The MilliAmpere is a small under-actuated passenger vessel, and the Northern Clipper is a large tanker vessel. We do not describe all simulation and ship parameters for every scenario. Hence, the full details on parameters are located in Appendix A.

4.1 Open-loop control scenario

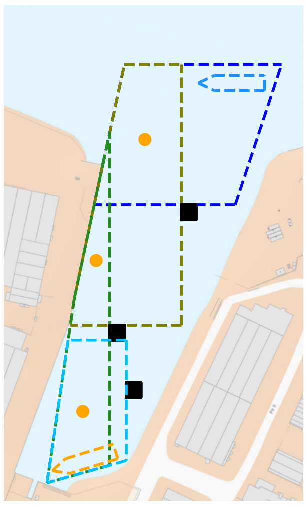
This section presents and discusses simulation results for methods A and B in an identical open-loop control scenario with the Northern Clipper model.

4.1.1 Method A

Method A has a horizon $T = 250\mathrm{s}$ where $N = 25$. Thus, giving the timestep $h = T/N = 10\mathrm{s}$. Figure 1a shows the planned path with four waypoints, including the dock pose (orange stapled line). The different colored stapled regions represent the convex spatial constraints the vessel must satisfy for each partitioned optimization problem. These constraints are what guarantee the vessel's safety when maneuvering. With the proposed solution of the constraints and waypoints, we intend to help the optimization avoid unwanted local optima. E.g., arrive at a position where obstacles make the vessel unable to proceed any closer to the dock pose.

Figure 1b shows the successful docking trajectory for method A with the large Northern Clipper model. At the end of the maneuver, the method showcases successful low-speed sway maneuvering to reach the dock pose. The entire problem is solved in 4.5 seconds, and each partitioned optimization problem in approximately 1.2 seconds. Such fast solution times indicate that the method has real-time feasibility.

The docking maneuver takes about 1000 seconds to perform in simulation, which is relatively slow. A significant portion comes from the vessel standing still because the vessel does full stops at every waypoint. Figure 4.1.2 best highlights this stop-and-go trajectory. The figure shows the states $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ of the vessel with extended periods of no positional change and zero velocity. Additionally, the figure shows spikes in velocity. In a practical sense, the stop-and-go trajectory may seem excessive and useless. However, we do not consider it to be a significant limiting factor as the implemented switching mechanism from Section 3.2.7 overcomes the problem in closed-loop control scenarios.


(1a) Method A's planned path. The waypoint and obstacle-free spatial constraints generation

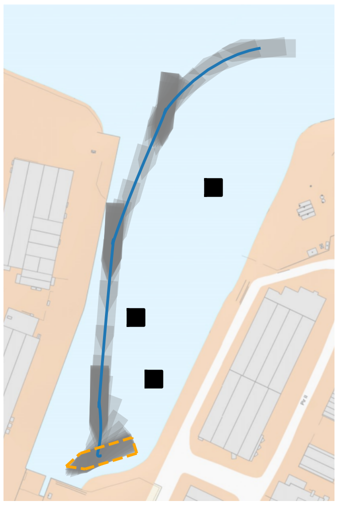

(1b) Method A's trajectory from the initial pose to the dock pose for the open-loop control scenario

Figure 4.1.1: Method A's planned waypoints and executed trajectory for the Northern Clipper in the open-loop control scenario

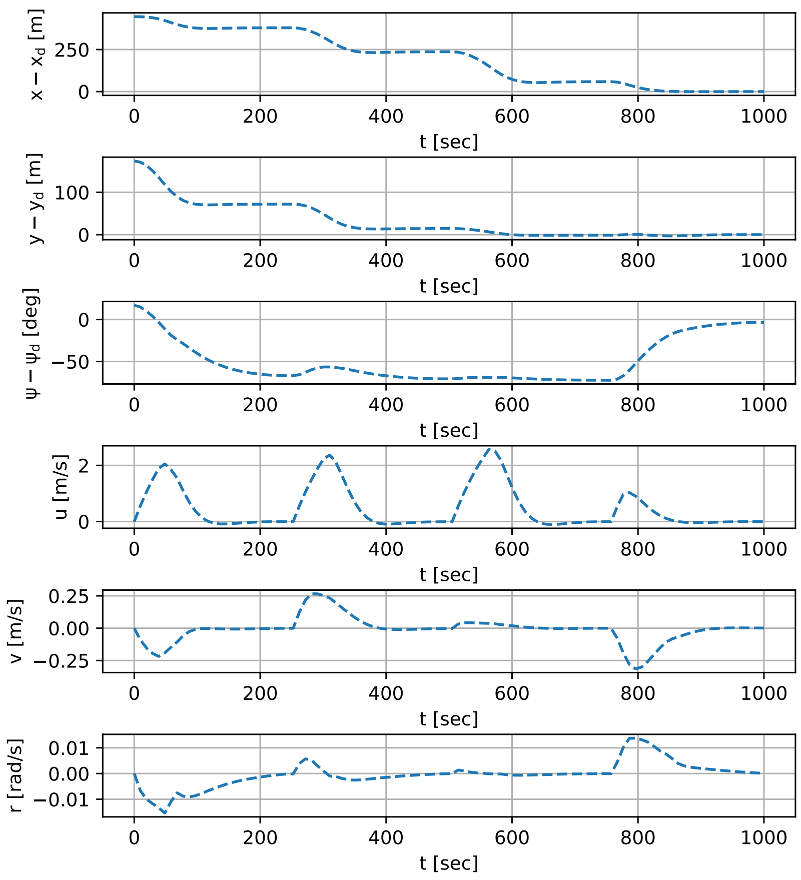


Figure 4.1.2: The state η and ν of method A's open-loop control scenario

4.1.2 Method B

This section presents the same scenario for method B, as discussed for method A in Section 4.1.1. Method B uses a longer horizon of $T = 350\text{s}$ where $N = 35$. Thus, giving the timestep $h = T/N = 10\text{s}$. Figure 4.1.3 shows the medial axis created from a constrained Delaunay triangulation. The medial axis provides a set of waypoints and the A* and waypoint reduction algorithms reduce the path, which leads to the initial guess in Figure 4a. The initial guess direct methods B's solution to a very different trajectory, shown in Figure 4b, than for method A (1b). By comparing the trajectory of method A (Figure 1b) and method B (Figure 4b), we can see how the waypoint generation and initial guess dictates the final solution for the respective methods.

When performing the docking maneuver, method B has a much smoother trajectory, as shown in Figure 4.1.5. The smooth trajectory also leads to a substantially faster dock maneuver of 350 seconds. The computation time of 19 seconds probably reflects the complexity of the optimization problem. A travel distance of 400 m before the vessel is supposed to initiate the low-speed maneuvers and come to a final stop along the quay creates a numerically complex optimization problem to calculate while satisfying the spatial and elliptic constraints.

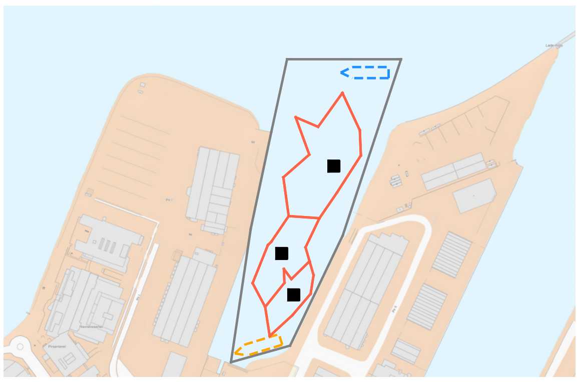
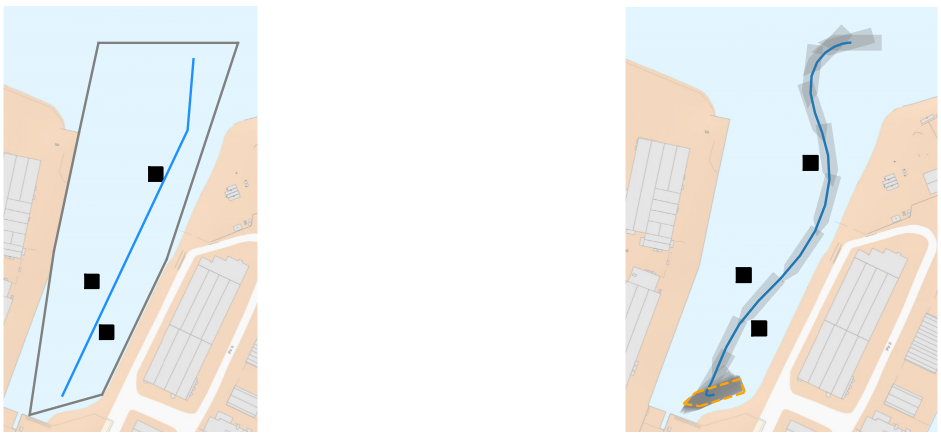


Figure 4.1.3: The medial axis of Method B's open-loop control scenario



(4a) The initial guess of method B's open-loop control scenario

(4b) Method B's complete trajectory for the open-loop control scenario

Figure 4.1.4: Method B's initial guess and executed trajectory for the Northern Clipper in the open-loop control scenario

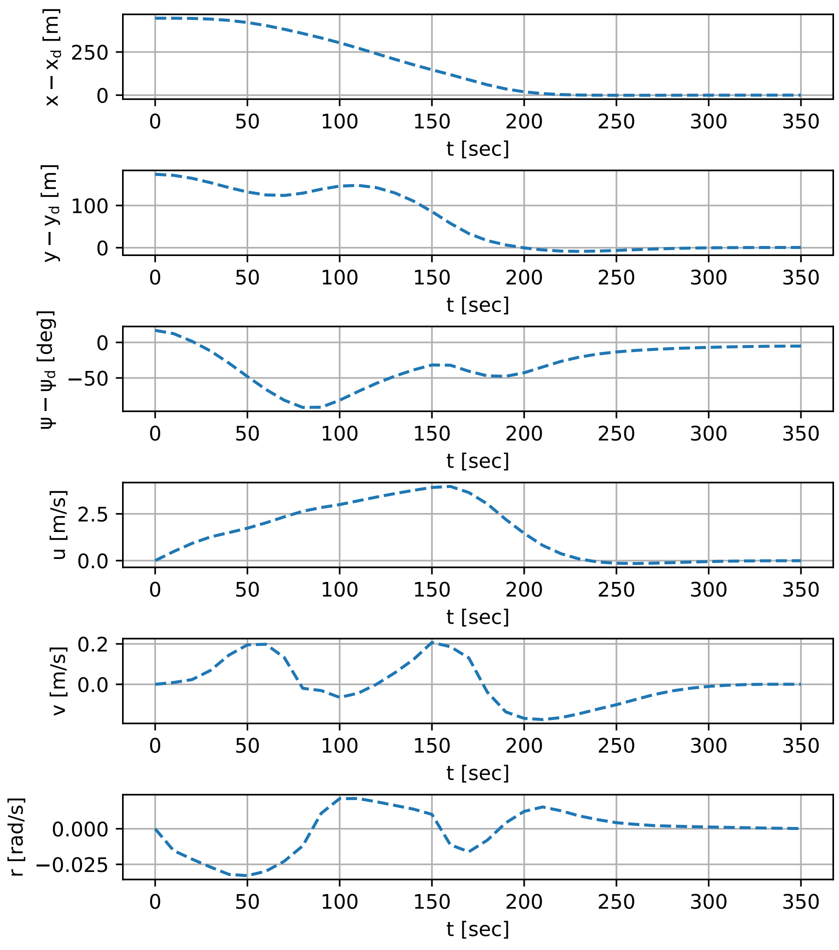


Figure 4.1.5: The state η and ν of method B's open-loop control scenario

4.2 Closed-loop control scenario

This section presents simulation results and the discussions of both methods in an identical closed-loop control scenario with the MilliAmpere vessel model. The optimization and control are separated as explained in Section 3.2.7.

4.2.1 Method A

This section presents the results for method A in a closed-loop control scenario. The planning horizon is $T_p = 120\text{s}$ where $N_p = 60$. Thus, giving the timestep $h_p = T_p/N_p = 2\text{s}$. The simulation and control horizon is $T_s = 500\text{s}$ where $N_s = 5000$, which gives the timestep $h_s = T_s/N_s = 0.1\text{s}$. The planner runs on 0.1Hz, planning a trajectory every 10 seconds, and the DP controller runs with a frequency of 10Hz.

Figure 4.2.1 shows the planned waypoints and spatial constraints of method A. A more direct and shorter path would likely be to travel between the two right-most obstacles. Our proposed method finds the shortest path with respect to the number of waypoints. Thus, it considers the planned path in 4.2.1 to be optimal. Figure 4.2.2 shows the vessel's planned (marked in orange) and executed (marked in blue) trajectory to the dock pose. Figure 4.2.2 indicates that the vessel has the same stop-and-go trajectory as in Section 4.1.1 despite using a more efficient waypoint switching mechanism. The proposed use of waypoints and switching makes it inevitable that the vessel might come to complete stops, and it does not entirely smooth out the planned path and trajectory. However, the extended periods of no positional change and zero velocity is completely absent in Figure 4.2.3 compared to Figure 4.1.2. We can still recognize a slight spike pattern in velocity in Figure 4.2.3. Still, this should not have any negative practical ramifications for docking.

When simulating with a separate planner and controller, method A performs remarkably better in docking time than in Section 4.1.1, primarily due to the faster switching. Figure 4.2.3 shows the vessel reaching the dock pose after approximately 280 seconds. With a horizon of 120 seconds and a partition into three optimization problems, we end up with a reduction of $120 \cdot 3 - 280 = 80\text{s}$ for the complete docking maneuver compared to running open-loop control. The reduction from faster switching grows with scaling the number of waypoints.

Method A shows excellent real-time performance. The optimization problem has an average solution time of 0.8s for the entire simulation, which is more than satisfactory for real-time performance where the optimization runs at 0.1Hz.

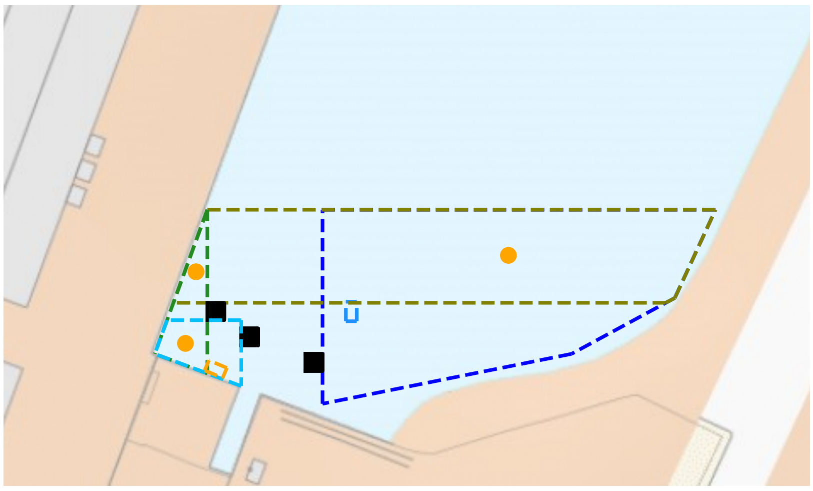


Figure 4.2.1: Method A's planned path. The waypoint and obstacle-free spatial constraints generation for the closed-loop control scenario

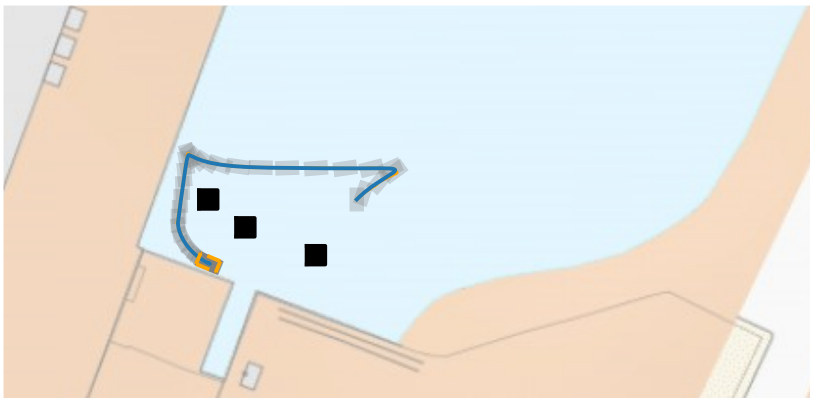


Figure 4.2.2: Method A's planned and executed trajectory from the initial pose to the dock pose for the closed-loop control scenario

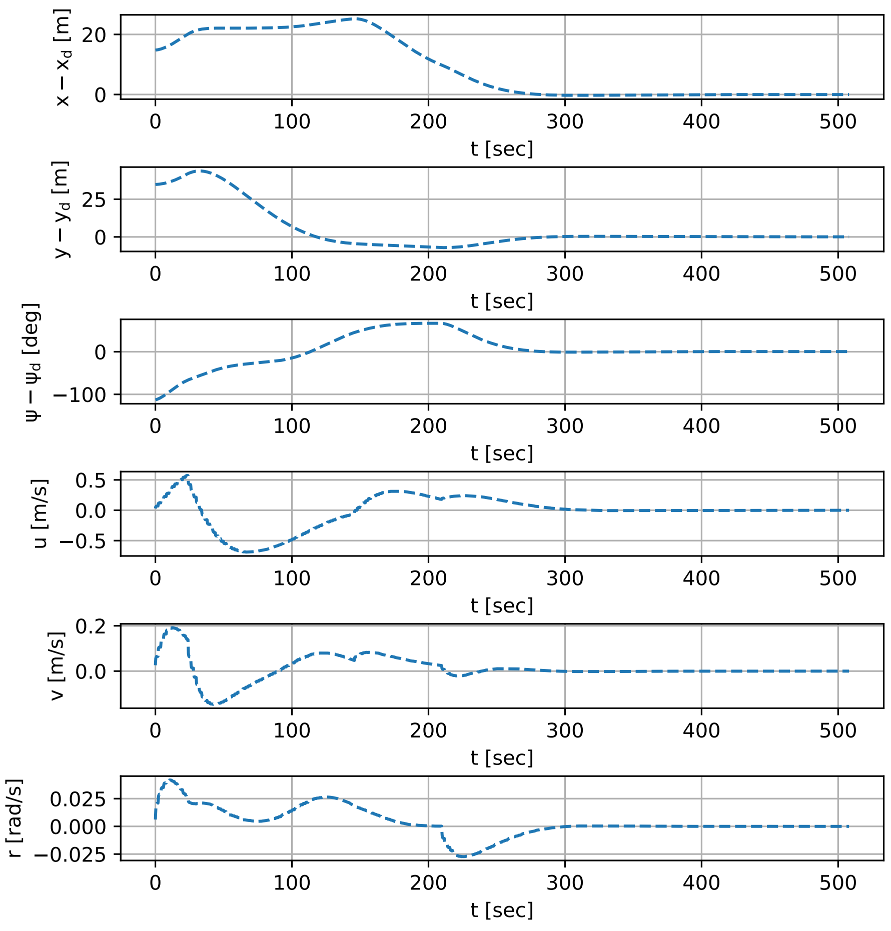


Figure 4.2.3: The state η and ν of method A's closed-loop control scenario

4.2.2 Method B

This section presents the results for method B in the same closed-loop control scenario. We set the simulation horizon to $T_s = 300\mathrm{s}$ where $N_s = 3000$, which gives the same timestep $h_s = T_s/N_s = 0.1\mathrm{s}$. The planner still runs on $0.1\mathrm{Hz}$, planning a trajectory every 10 seconds, and the DP controller runs with a frequency of $10\mathrm{Hz}$.

The initial guess of method B is shown in Figure 4.2.4, choosing a shorter and more direct path to the dock pose than method A. Figure 4.2.5 shows how the vessel accurately follows the planned trajectory (marked in orange) from the initial pose to the final dock pose. We also see a much smoother trajectory in Figure 4.2.6 than in Figure 4.2.3.

Using the Pseudo-Huber objective function in the optimization, method B significantly improves the computational performance. Such a considerable boost in computational efficiency is likely due to shorter travel distances and linear penalization when further away from the dock pose. A pseudo-Huber objective function allows faster computation and more numerical stability when further away from the dock pose (Gros and Zanon (2017)).

In this closed-loop control scenario, method B solves the optimization problem at an average of 1.2s. Such fast computation also enables method B for real-time performance, competing with the computational speed of method A. Method B still outperforms method A in terms of docking speed and human-like behavior, reaching the dock pose approximately 100 seconds earlier. In a practical scenario, docking in 300 or 200 seconds might not make a significant enough difference in choosing one method over another. However, the difference between the two methods will likely increase as we increase the travel time of the vessel. E.g., as we see in Section 4.1.1, the difference in docking time is close to 700 seconds. Such a large disparity would likely have a bigger influence on which approach is the most suitable.

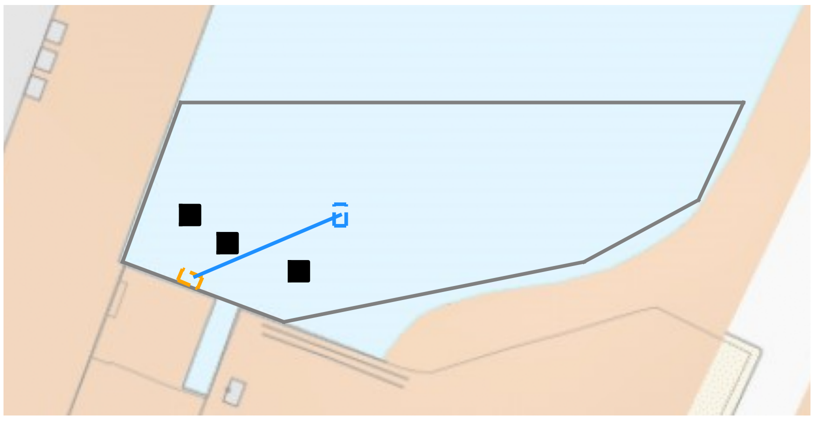


Figure 4.2.4: The initial guess of method B's closed-loop control scenario

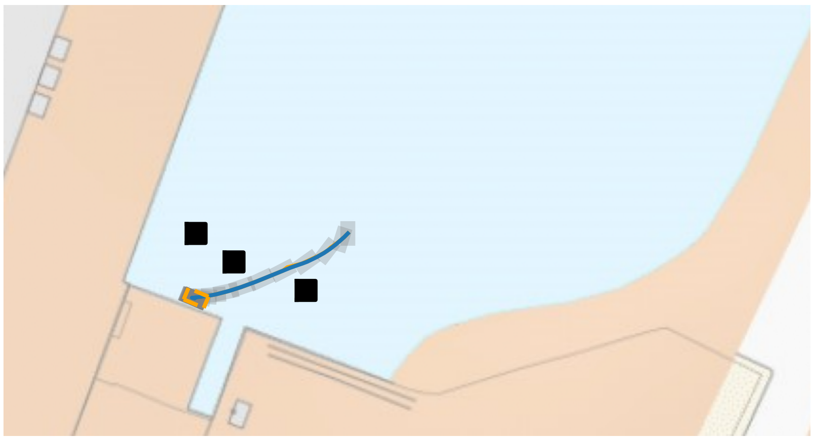


Figure 4.2.5: Method B's planned and executed trajectory from the initial pose to the dock pose for the closed-loop control scenario

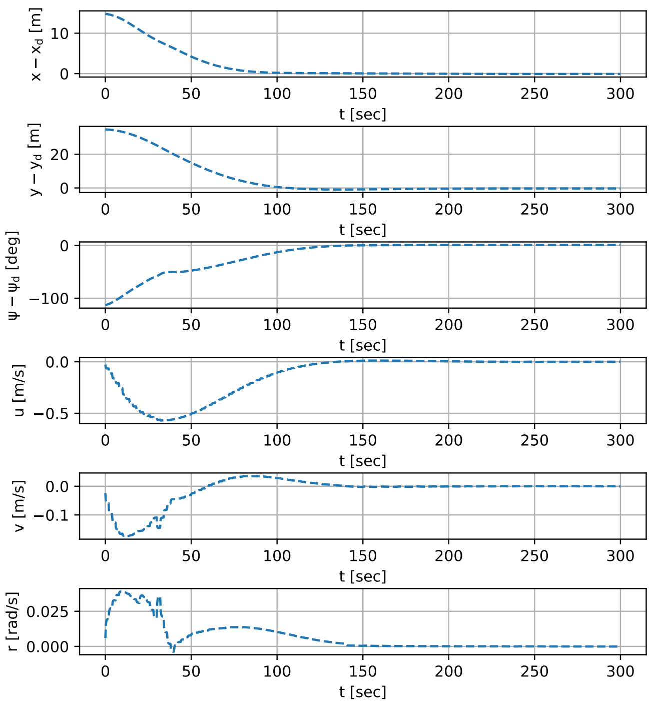


Figure 4.2.6: The state η and ν of method B's closed-loop control scenario

4.3 Partially known map and dynamic obstacle scenario

This section presents results for a Partially known map and dynamic obstacle scenario. Both methods initially plan based on knowledge of two obstacles obstructing the direct path to the dock pose. We assume the vessel has a sensor that detects objects at 20m. After some time, the vessel detects a new obstacle that is supposed to block the initially planned path, forcing the methods to replan a new path. When the vessel tracks the new path, it will do the second detection of a moving obstacle and execute a simple stop-and-wait maneuver until the moving obstacle has passed. This scenario shows how the methods easily pair with more sophisticated collision avoidance systems and how they have practical viability in real-life scenarios where it is essential to react to a dynamic environment.

4.3.1 Method A

This section presents the results for method A in a Partially known map and dynamic obstacle scenario. The planning horizon is $T_p = 120\text{s}$ where $N_p = 60$. Thus, giving the timestep $h_p = T_p/N_p = 2\text{s}$. The simulation horizon is $T_s = 600\text{s}$ and $N_s = 6000$, which gives the timestep $h_s = T_s/N_s = 0.1\text{s}$. The dynamic obstacle is moving with a velocity of 1m/s to ensure that the vessel has to do a complete stop and wait for the obstacle to pass before it can continue the docking procedure.

Figure 4.3.1 shows the initial map and the waypoints planned. Figure 4.3.2 shows how the detection of a new obstacle (marked in red) forces a replan of the path and the vessel to do a full turnaround to reach the dock pose.

Figure 4.3.3 shows the vessel's complete docking trajectory from the initial pose to the dock pose. Figure 3a shows the planned (marked in orange) and executed (marked in blue) docking trajectory until the vessel detects the first obstacle. The vessel is initially planning to maneuver through the area where the new obstacle appears and has to do a replan and complete turnaround to reach the dock pose. Figure 3b shows the vessel turning around after detecting the first obstacle and the resulting planned and executed trajectory until it detects a new, moving obstacle. At this point in time, the vessel starts to slow down and completely stop until the moving obstacle has moved out of range. Figure 3b also shows how our method helps avoid local minima by forcing the vessel to do a complete turnaround with a replanned path. Figure 3c shows the vessel's trajectory until it passes the prior position of the moving obstacle and successfully avoids the collision. Figure 3d shows the vessel's complete planned and executed docking trajectory, successfully reaching the dock pose.

Figure 4.3.4 shows the state η and ν of the vessel and that it takes the vessel approximately 350 seconds to dock successfully. This figure highlights well the effect of the switching mechanism. We recognize the same spike pattern in velocity in Figure 4.3.4 as in Figure 4.1.2. However, in Figure 4.3.4, we still avoid the extended periods of no zero velocity, which considerably speeds up the docking procedure.

The computation time still indicates good real-time performance. The average solution time of the optimization is 0.7 seconds, maintaining a good margin to the 10 second planning horizon.

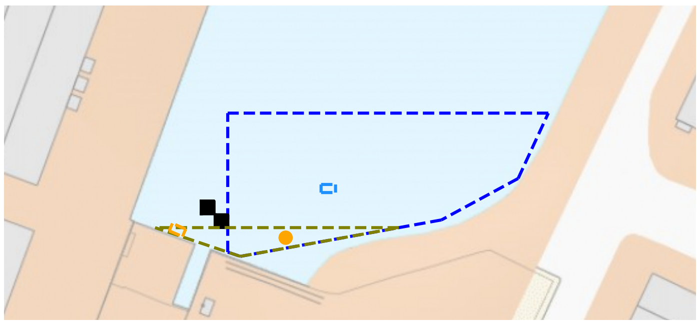


Figure 4.3.1: Method A's initial planned path for the Partially known map and dynamic obstacle scenario. Before the first obstacle was detected

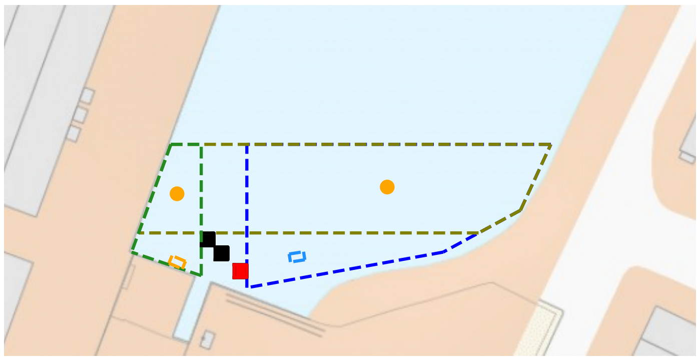
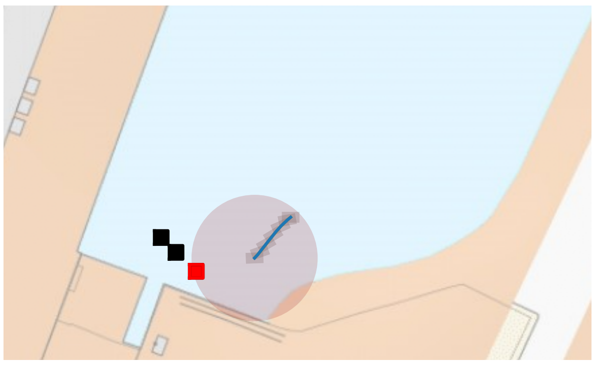
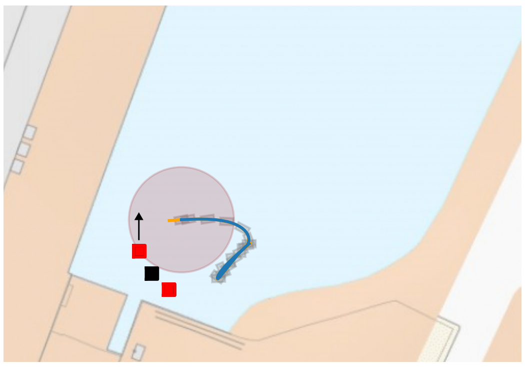


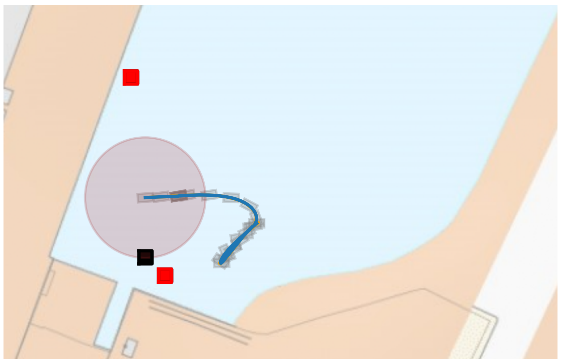
Figure 4.3.2: Method A's replanned path for the Partially known map and dynamic obstacle scenario. After detecting a new obstacle (marked in red)



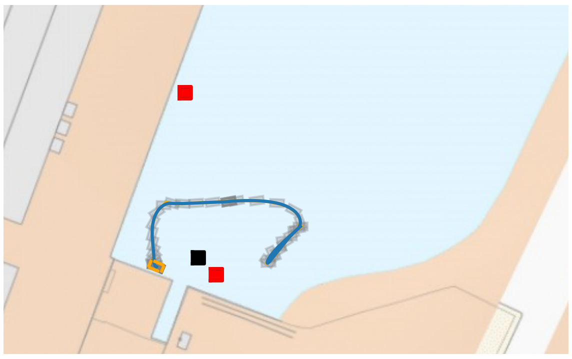
(3a) Method A's planned and executed trajectory from the initial pose until the vessel detects the first obstacle (marked in red)



(3b) Method A's planned and executed trajectory from the initial pose until the vessel detects the moving obstacle and starts slowing down



(3c) Method A's planned and executed trajectory from the initial pose to the pose where the vessel is passing the interfering obstacle



(3d) Method A's planned and executed trajectory from the initial pose to dock pose with initially untracked obstacles marked in red

Figure 4.3.3: Method A's planned and executed trajectory from the initial pose to the dock pose for the Partially known map and dynamic obstacle scenario

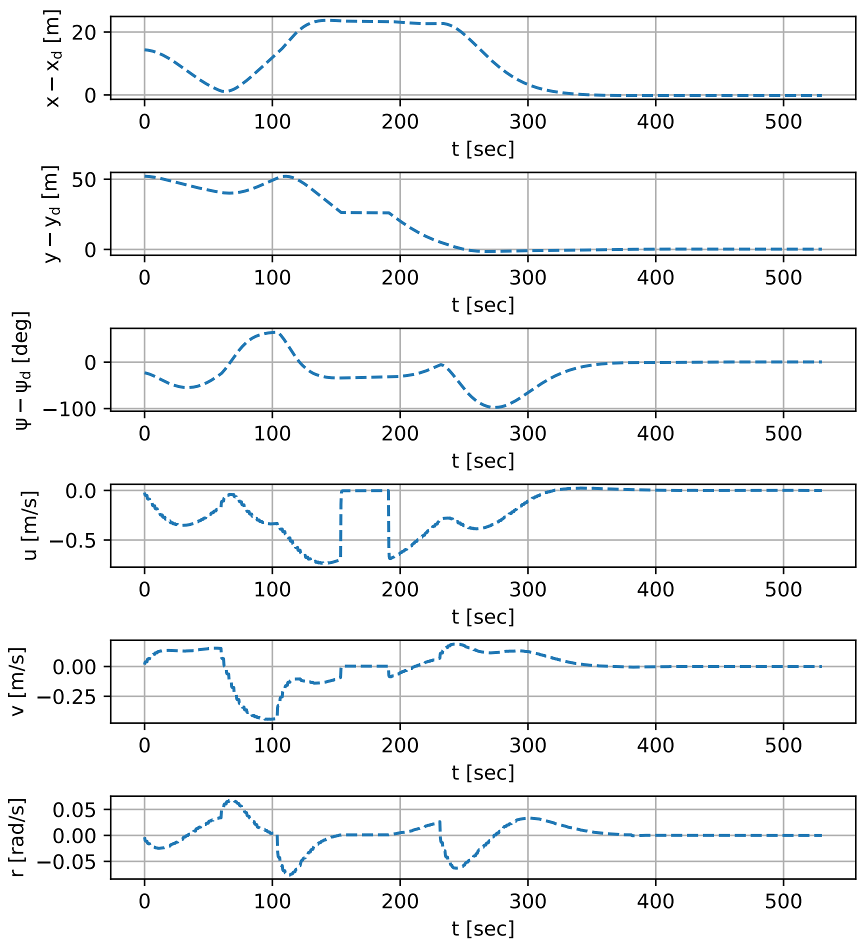


Figure 4.3.4: The state η and ν of method A in the Partially known map and dynamic obstacle scenario

4.3.2 Method B

This section presents the results for method B in a Partially known map and dynamic obstacle scenario. The planning and simulation horizon is similar to Section 4.3.1. The dynamic obstacle is moving with the same velocity of $1 \mathrm{m/s}$.

Figure 4.3.5 shows the initial guess of the method at time $T = 0\mathrm{s}$. The replanned path we show in Figure 4.3.6 is when detecting the unknown obstacle (marked in red) obstructing the initially planned path to the dock pose. Figure 4.3.7 shows the planned (marked in orange) and executed (marked in blue) trajectory. Figure 4.3.8 shows the state η and ν.

Figure 7a shows the planned and executed trajectory of the vessel until the vessel detects the moving obstacle and starts slowing down to stop. One can also see the vessel's heading in the initial phase of the trajectory facing in the direction of the initially planned path before the vessel does a turn and redirects to the replanned path. Figure 7b shows the planned and executed trajectory until the vessel passes the prior position of the dynamic obstacle, sufficiently clearing the obstacle. The darker grey vessel hulls show the pose of the vessel when it stops and waits for the dynamic obstacle to pass. The complete planned and executed docking trajectory is shown in Figure 7c, where the vessel successfully reaches the dock pose.

In this scenario, the vessel continues in the same direction but has to make a minor heading adjustment to track the new path. One could argue that this scenario is less complex than the one for method A because the vessel does not have to make any drastic maneuvers to change its course after replanning. However, Figure 4.3.7 shows that method B can account for new information, which is critical for practical viability. The computational speed of method B still shows good real-time performance, with an average solution time of 1.2 seconds for the optimization planning stage.

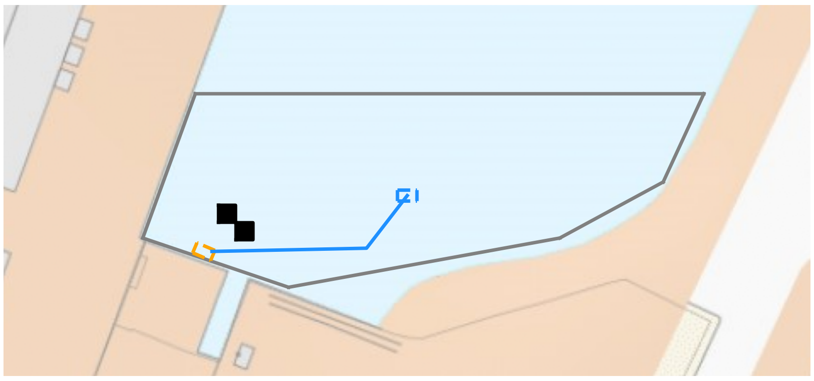


Figure 4.3.5: Method B's initial guess for the Partially known map and dynamic obstacle scenario. Before the first obstacle was detected

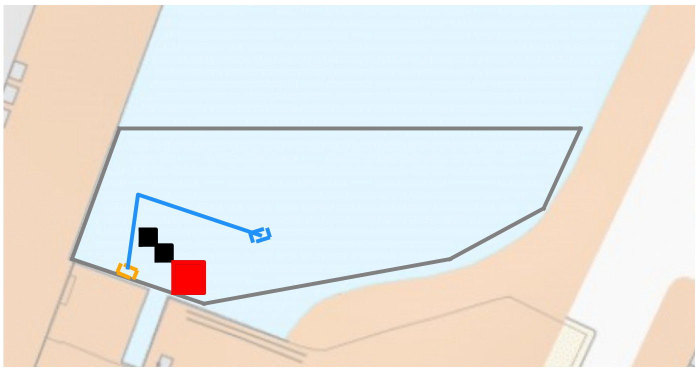
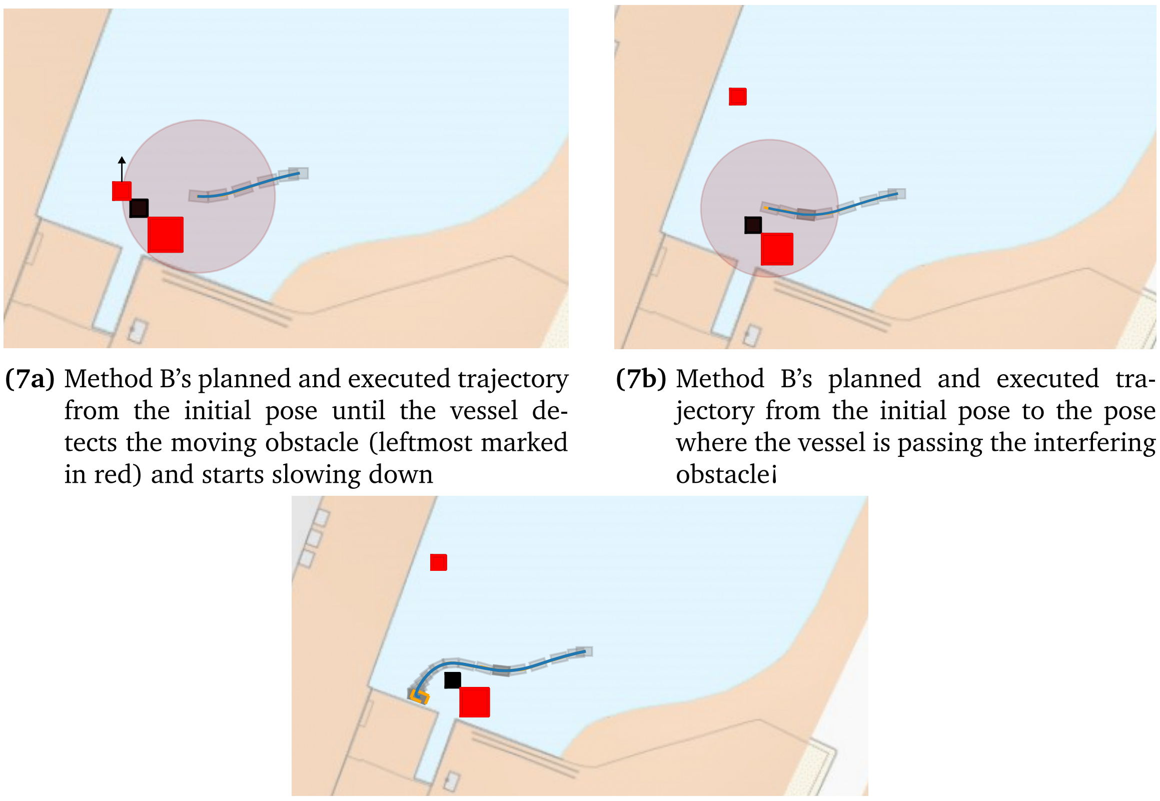


Figure 4.3.6: Method B's replanned initial guess for the Partially known map and dynamic obstacle scenario after detecting the first obstacle (marked in red)



(7a) Method B's planned and executed trajectory from the initial pose until the vessel detects the moving obstacle (leftmost marked in red) and starts slowing down

(7b) Method B's planned and executed trajectory from the initial pose to the pose where the vessel is passing the interfering obstaclei

(7c) Method B's planned and executed trajectory from the initial pose to dock pose with initially untracked obstacles marked in red

Figure 4.3.7: Method B's planned and executed trajectory from the initial pose to the dock pose for the Partially known map and dynamic obstacle scenario

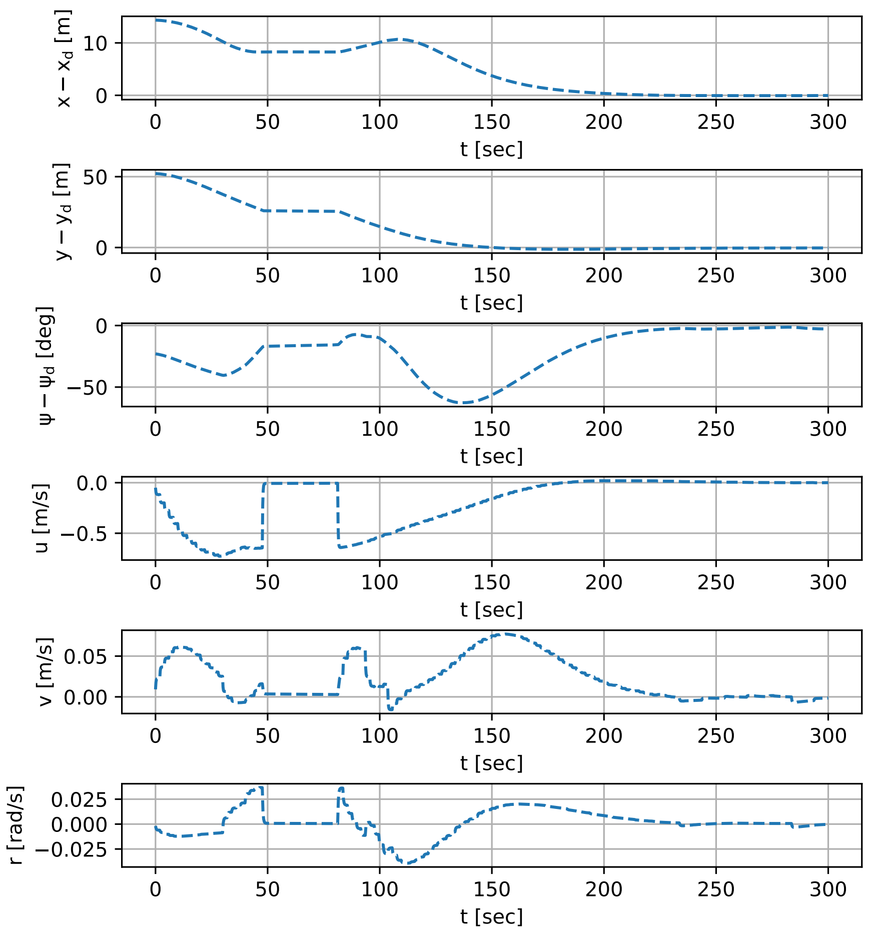


Figure 4.3.8: The state η and ν of method B in the Partially known map and dynamic obstacle scenario

4.4 Limitations

In order to better highlight the strength and weaknesses of the two methods, Figure 4.4.1 and Figure 4.4.3 show scenarios where the methods can not find feasible solutions. Let us consider the scenario of Figure 4.4.1 where method A fails to find a feasible solution. Initially, the trajectory of method B seems remarkably simple and easy to execute. The position of the obstacles makes it not as simple to solve for method A. Method A finds a decomposition and feasible path. However, the overlapping areas are too small for the vessel to maneuver and turn within, as illustrated in Figure 4.4.2. With such a large vessel, the placement of the obstacles creates a worst-case scenario.

When the vessel can not freely maneuver, the vessel has a considerably higher chance of achieving undesirable positional states. Undesirable positional states complicate further guidance and control, which ultimately leads to the optimization not being able to find a feasible solution. The geometry of the harbor and obstacles highly influence the feasibility of method A, which is the method's advantage and limitation. The advantage is that the exploitation of the harbor geometry helps the vessel avoid unwanted local optima docking next to obstacles, which many optimization-based methods would struggle to avoid. The limitation emerges when the geometry of the harbor creates a decomposition which leads to an infeasible optimization problem. Method B shows how effortless the problem in Figure 4.4.1 can be solved. Which also showcases the efficient, smooth, and human-like maneuvering one can achieve with method B.

Method B is more susceptible than method A to getting stuck in local optima next to obstacles and never reaching the dock pose. For instance, in the scenario in Figure 4.4.3, method B finds a local optimum next to an obstacle and never reaches the dock pose. The vessel stops at a local optimum partly because the initial guess does not account for the vessel's hull and warm-starts the optimization with an infeasible path. However, getting stuck in a local optimum is primarily because we do non-linear optimization. Thus, it is unavoidable in some cases. Our proposed solution in Method A is designed to overcome this exact challenge. In many scenarios where other optimization-based methods fail to find a solution, our proposed method A will force the vessel to pass specific waypoints and find a feasible solution.

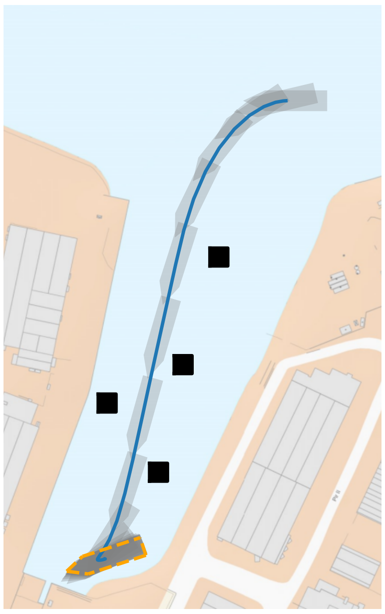


Figure 4.4.1: Method B's trajectory of an open-loop scenario where method A does not find a feasible solution

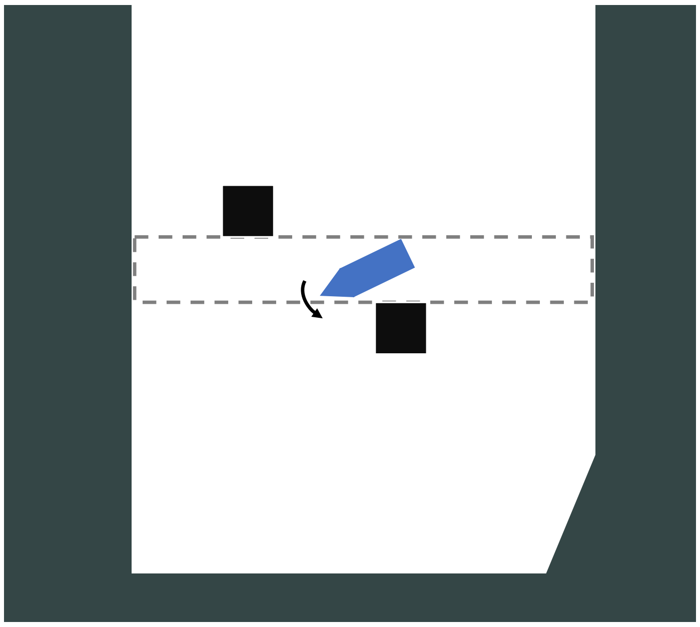


Figure 4.4.2: Method A's worst-case scenario where the vessel is unable to turn inside a decomposed region

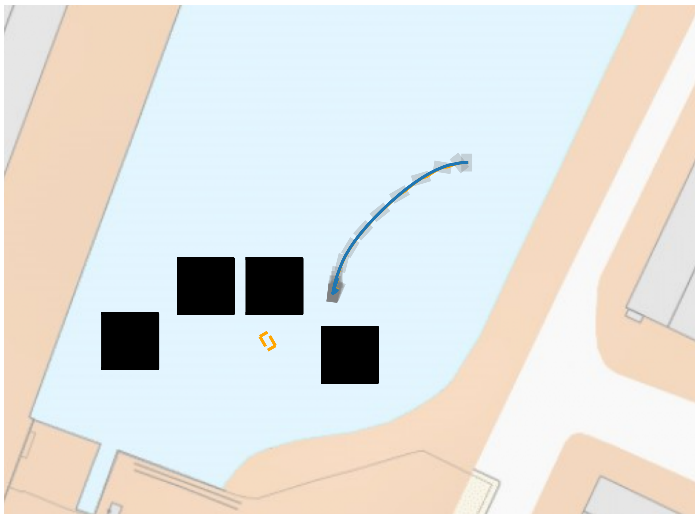


Figure 4.4.3: Method B's trajectory where the vessel gets stuck in a local optima

Chapter 5

Conclusion and future work

Our work presented in this thesis proposes two methods combining computational geometry with numerical optimal control to do static and dynamic obstacle avoidance and successfully dock a vessel. Both methods show good performance for all scenarios presented; this includes complex and challenging docking scenarios with open-loop control, closed-loop control, and a partially-known environment with a dynamic obstacle. The methods are also demonstrating good performance for two distinctive vessel models, which showcases their robustness for dealing with changing model dynamics. The ability to handle environments where all information is not readily available shows a great deal of robustness and reliability in terms of collision avoidance and safety. They both can successfully replan and track an entirely new path. Thus, accounting for newly detected obstacles on the fly while keeping the vessel safe. Additionally, we show that pairing them with more comprehensive and sophisticated collision avoidance systems is a straightforward task. The ability to account for new information on the fly combined with real-time computational feasibility indicates that the methods show promising results for practical use.

Method A utilizes obstacle tangents to do a tangential decomposition of a non-convex polygon. From the decomposition, we create a graph where the overlapping areas form the edges connecting the decomposed, convex polygons, and the A* finds the optimal path of polygons to traverse. Lastly, the optimization finds a feasible obstacle-free docking trajectory connecting the waypoints. Method B uses constrained Delaunay triangulation to form a polygon's medial axis and create a graph connecting the medial axis to the initial and dock pose. The A* and waypoint reduction algorithm constructs the straight-lined initial guess used for warm-starting an optimization problem that finds a feasible obstacle-free docking trajectory.

The decomposition of method A is an extension and improved solution of the work by Martinsen et al. (2019) and Ødven (2021). We have implemented weights that prefer larger overlapping areas to help avoid the worst-case scenarios discussed in Section 4.4. Additionally, we have improved the decomposition to compute the various convex polygons constrained by the opposite facing tangents instead of finding just one decomposed polygon. The method still heavily relies on the geometry of the docking area and obstacles. As shown in Figure 4.4.1, scenarios where the vessel no longer can freely maneuver inside the convex constraints makes it difficult to find a feasible solution. The problem of too small convex constraints will most likely increase when scaling the number of obstacles present. The method also tends to find less optimal solutions from a human perspective. Less optimal solutions from a human perspective might not be a significant problem in practice, as it is more important that we can reach the dock pose rather than having the most natural and efficient path. The extension implemented for method A makes it possible for our proposed solution to avoid multiple obstacles simultaneously. Additionally, we show in Chapter 4.3 that it handles map inaccuracies and can account for dynamic obstacles.

For method B, as stated in Bitar et al. (2019), the optimization will lock onto one route choice due to it being warm started. Therefore, a reasonable initial guess for whatever goal one wants to achieve is vital. In our case, the initial guess could dictate whether the optimization finds a feasible solution or not. E.g., as the initial guess of method B does not account for the vessel's hull, the vessel could end up in a similar situation as in Figure 4.4.3. With a good initial guess, there is still no guarantee that the optimization avoids local optima and successfully docks.

The complexity of the optimization problem is another factor why the initial guess is of such importance. Method B has a higher complexity due to more spatial constraints, and as we show in Section 4.1.2, the optimization takes quite a while to compute compared to method A. An initial guess that accounts for vessel dynamics could severely improve the run time. Even though we showed that method B also was feasible for real-time performance, it is worth noting that improvements revolving the initial guess could benefit the computation time and counteract the challenges related to the complexity of the problem. The vessel's safety is also dependent on the complexity of the optimization. Our optimization problem can guarantee that the sampled points of the vessel hull will avoid a collision. In some cases, e.g., with a sparse sample of the hull we can not guarantee a collision-free trajectory for a large vessel. A denser sample will help solve such problems, but also adds to the complexity of the optimization.

The methods showcase different strengths and weaknesses, and as we have discovered throughout the work in this thesis, they complement each other in ways that might make it efficient to combine the two methods in future work. Method B has more time-efficient human-like trajectories. Method A tends to avoid local optima better and perform better in geometrically complex docking scenarios. Thus, a combination where one could exploit the strengths of both methods could be natural to investigate. The combination could be to merge the two methods and create a hybrid approach incorporating both of their strengths. With the computational speed of the methods, they could even be run as a redundant system. E.g., we would prefer method B as long as it finds a feasible solution and does not get stuck in local optima. Whenever one of the two occurs, we can make switch to method A.

It could also be natural to investigate improvements for both methods individually. Future work could include making the decomposition more efficient, leading to faster computations for method A. Keeping the same tangential decomposition methodology with triangular obstacle modeling would be interesting to investigate. That approach could contribute to fewer waypoints and larger overlapping regions and potentially help minimize the limitations of the methods. For method B, a natural focus would be the initial guess and how we can alter it to account for the vessel's hull and dynamics. Other areas of improvement could be to incorporate some of the ideas from method A to have better performance when facing scenarios where the vessel otherwise would not reach the dock pose.

Bibliography

Optimization on planning of trajectory and control of autonomous berthing and unberthing for the realistic port geometry. *Ocean Engineering*, 245:110390, 2022. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2021.110390. URL `https://www.sciencedirect.com/science/article/pii/S0029801821016826`.

Joel Andersson, Joris Gillis, and Greg Horn. *Welcome to CasADi's documentation!*, 2021. `https://web.casadi.org/docs/` [Accessed: 2021-12-16].

Seungdae Baek and Joohyun Woo. Model reference adaptive control-based autonomous berthing of an unmanned surface vehicle under environmental disturbance. *Machines*, 10(4), 2022. ISSN 2075-1702. doi: 10.3390/machines10040244. URL `https://www.mdpi.com/2075-1702/10/4/244`.

Kristoffer Bergman, Oskar Ljungqvist, Jonas Linder, and Daniel Axehill. An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5283–5290, 2020. doi: 10.1109/CDC42340.2020.9303746.

Glenn Bitar, Vegard N. Vestad, Anastasios M. Lekkas, and Morten Breivik. Warm-started optimized trajectory planning for asvs. *IFAC-PapersOnLine*, 52(21): 308–314, 2019. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2019.12.325. URL `https://www.sciencedirect.com/science/article/pii/S2405896319322104`. 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2019.

Glenn Bitar, Andreas B. Martinsen, Anastasios M. Lekkas, and Morten Breivik. Trajectory planning and control for automatic docking of asvs with full-scale experiments. *IFAC-PapersOnLine*, 53(2):14488–14494, 2020. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2020.12.1451. URL `https://www.sciencedirect.com/science/article/pii/S2405896320318632`. 21st IFAC World Congress.

Glenn Bitar, Bjørn-Olav H. Eriksen, Anastasios M. Lekkas, and Morten Breivik. Three-phase automatic crossing for a passenger ferry with field trials. In *2021 European Control Conference (ECC)*, pages 2271–2277, 2021. doi: 10.23919/ECC54610.2021.9655139.

Morten Breivik and Jon-Erik Loberg. A virtual target-based underway docking procedure for unmanned surface vehicles. *IFAC Proceedings Volumes*, 44 (1):13630–13635, 2011. ISSN 1474-6670. doi: https://doi.org/10.3182/20110828-6-IT-1002.02969. URL `https://www.sciencedirect.com/science/article/pii/S1474667016458143`. 18th IFAC World Congress.

Edmund F. Brekke, Egil Eide, Bjørn-Olav H. Eriksen, Erik F. Wilthil, Morten Breivik, Even Skjellaug, Øystein K. Helgesen, Anastasios Lekkas, Andreas B. Martinsen, Emil H. Thyri, Tobias Torben, Erik Veitch, Ole A. Alsos, and Tor Arne Johansen. milliampere: An autonomous ferry prototype. 2022. URL `https://folk.ntnu.no/edmundfo/papers/icmass-milliampere-2022.pdf`.

Thomas J. Böhme and Benjamin Frank. *Indirect Methods for Optimal Control*, pages 215–231. Springer International Publishing, 2017. ISBN 978-3-319-51317-1. doi: 10.1007/978-3-319-51317-1_7. URL `https://doi.org/10.1007/978-3-319-51317-1_7`.

Wenqi Cai, Arash B. Kordabad, Hossein N. Esfahani, Anastasios M. Lekkas, and Sébastien Gros. Mpc-based reinforcement learning for a simplified freight mission of autonomous surface vehicles. In *2021 60th IEEE Conference on Decision and Control (CDC)*, page 2990–2995. IEEE Press, 2021. doi: 10.1109/CDC45484.2021.9683750. URL `https://doi.org/10.1109/CDC45484.2021.9683750`.

Kuang-Hua Chang. Chapter 2 - structural analysis. In *Product Performance Evaluation with CAD/CAE*, The Computer Aided Engineering Design Series, pages 43–119. Academic Press, Boston, 2013. ISBN 978-0-12-398460-9. doi: https://doi.org/10.1016/B978-0-12-398460-9.00002-0. URL `https://www.sciencedirect.com/science/article/pii/B9780123984609000020`.

Moritz Diehl and Sébastien Gros. *Numerical Optimal Control - script draft*. 2017. URL `https://www.syscop.de/files/2020ss/NOC/book-NOCSE.pdf`.

Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, 2011. ISBN 9781119994138.

Thor I. Fossen. *MSS Toolbox*, 2021. `https://github.com/cybergalactic/MSS` [Accessed: 2021-12-16].

T.I. Fossen, S.I. Sagatun, and A.J. Sørensen. Identification of dynamically positioned ships. *Control Engineering Practice*, 4(3):369–376, 1996. ISSN 0967-0661. doi: https://doi.org/10.1016/0967-0661(96)00014-7. URL `https://www.sciencedirect.com/science/article/pii/0967066196000147`.

Sean Gillies. *The Shapely User Manual*, 2021. `https://shapely.readthedocs.io/en/stable/manual.html`[Accessed:2021-12-16].

Sébastien Gros and Moritz Diehl. Nmpc based on huber penalty functions to handle large deviations of quadrature states. In *2013 American Control Conference*, pages 3159–3164, 2013. doi: 10.1109/ACC.2013.6580317.

Sébastien Gros and Mario Zanon. Penalty functions for handling large deviation of quadrature states in nmpc. *IEEE Transactions on Automatic Control*, 62(8): 3848–3860, 2017. doi: 10.1109/TAC.2017.2649043.

Ella-Lovise Hammervold. Automatic docking of an autonomous surface vessel. Master's thesis, Norwegian University of Science and Technology (NTNU), 2020. URL `https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2656724/no.ntnu%3Ainspera%3A47198187%3A16519857.pdf?sequence=1&isAllowed=y`.

Matanya B. Horowitz, Anil Damle, and Joel W. Burdick. Linear hamilton jacobi bellman equations in high dimensions. In *53rd IEEE Conference on Decision and Control*, pages 5880–5887, 2014. doi: 10.1109/CDC.2014.7040310.

HSL. *A collection of Fortran codes for large-scale scientific computation*, 2022. `http://www.hsl.rl.ac.uk` [Accessed: 2022-05-30].

C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder. Efficient direct multiple shooting for nonlinear model predictive control on long horizons. *Journal of Process Control*, 22(3):540–550, 2012. ISSN 0959-1524. doi: https://doi.org/10.1016/j.jprocont.2012.01.008. URL `https://www.sciencedirect.com/science/article/pii/S095915241200011X`.

Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. 1998.

D. T. Lee. Proximity and reachibility in the plane. Technical report, University of Illinois, Coordinated Science Laboratory, 1978.

Simon Leonard, Kyle Wu, Yonjae Kim, Axel Krieger, and Peter Kim. Smart tissue anastomosis robot (star): A vision-guided robotics system for laparoscopic suturing. *Biomedical Engineering, IEEE Transactions on*, 61:1305–1317, 04 2014. doi: 10.1109/TBME.2014.2302385.

Shijie Li, Jialun Liu, Rudy R Negenborn, and Qing Wu. Automatic docking for underactuated ships based on multi-objective nonlinear model predictive control. *IEEE Access*, 8:70044–70057, 2020.

Andreas B. Martinsen, Anastasios M. Lekkas, and Sebastien Gros. Autonomous docking using direct optimal control. *IFAC-PapersOnLine*, 52(21): 97–102, 2019. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2019.12.290. URL `https://www.sciencedirect.com/science/article/pii/S2405896319321755`. 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2019.

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. *IEEE Access*, 8:204974–204986, 2020. doi: 10.1109/ACCESS.2020.3037171.

The matplotlib development team. *Matplotlib 3.5.1 documentation*, 2021. `https://matplotlib.org/stable/index.html` [Accessed: 2021-12-16].

Travis E. Oliphant. *Guide to NumPy*, 2021. `https://numpy.org/doc/stable/` [Accessed: 2021-12-16].

Anders Aglen Pedersen. Optimization based system identification for the milliampere ferry. Master's thesis, Norwegian University of Science and Technology (NTNU), 2019. URL `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2625699`.

Dimas M. Rachman, Atsuo Maki, Yoshiki Miyauchi, and Naoya Umeda. Warm-started semionline trajectory planner for ship's automatic docking (berthing). *Ocean Engineering*, 252:111127, 2022. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2022.111127. URL `https://www.sciencedirect.com/science/article/pii/S0029801822005388`.

G.J.S. Rae, S.M. Smith, D.T. Anderson, and A.M. Shein. A fuzzy rule based docking procedure for two moving autonomous underwater vehicles. In *1993 American Control Conference*, pages 580–584, 1993. doi: 10.23919/ACC.1993.4792923.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

Ryohei Sawada, Koichi Hirata, Yasushi Kitagawa, Eiko Saito, Michio Ueno, Katsuji Tanizawa, and Junji Fukuto. Path following algorithm application to automatic berthing control. *Journal of Marine Science and Technology*, 26,

2020. doi: 10.1007/s00773-020-00758-x. URL https://doi.org/10.1007/s00773-020-00758-x.

The sciPy community. *SciPy documentation*, 2021. https://scipy.github.io/devdocs/index.html [Accessed: 2021-12-16].

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. *Diagnostic pathology*, 7:134, 10 2012. doi: 10.1186/1746-1596-7-134. URL https://doi.org/10.1186/1746-1596-7-134.

Yonghui Shuai, Guoyuan Li, Xu Cheng, Robert Skulstad, Jinshan Xu, Honghai Liu, and Houxiang Zhang. An efficient neural-network based approach to automatic ship docking. *Ocean Engineering*, 191:106514, 2019. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2019.106514. URL https://www.sciencedirect.com/science/article/pii/S0029801819306535.

Henrik Bjering Strand. Autonomous docking control system for the otter usv: A machine learning approach. Master's thesis, Norwegian University of Science and Technology (NTNU), 2020. URL https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2780950.

Tesla. *Artificial Intelligence & Autopilot*, 2022. https://www.tesla.com/AI[Accessed:2022-06-01].

Per Gunnar Berg Torvund. Nonlinear autonomous docking and path-following control systems for the otter usv. Master's thesis, Norwegian University of Science and Technology (NTNU), 2020. URL https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2780866.

H. Yamato. Automatic berthing by the neural controller. *Proc. of Ninth Ship Control Systems Symposium*, 3:3183–3201, 1990. URL https://ci.nii.ac.jp/naid/10010732178/en/.

Yusi Zhou, Nailong Wu, Haodong Yuan, Feng Pan, Zhiyong Shan, and Chao Wu. Pde formation and iterative docking control of usvs for the straight-line-shaped mission. *Journal of Marine Science and Engineering*, 10(4), 2022. ISSN 2077-1312. doi: 10.3390/jmse10040478. URL https://www.mdpi.com/2077-1312/10/4/478.

Petter Ødven, Andreas B. Martinsen, and Anastasios M. Lekkas. Static obstacle avoidance and docking of asvs using computational geometry and numerical optimal control. 2022. Accepted for the 14th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2022.

Petter K. Ødven. Docking and static obstacle avoidance for an asv using optimal control and computational geometry. Project report in TTK4550, Department of Engineering Cybernetics, NTNU – Norwegian University of Science and Technology, 2021.

Appendix A

Simulation and optimization parameters

The parameters for the MilliAmpere, Northern Clipper and optimization problem are given in Table A.0.1, Table A.0.2, and Table A.0.3, respectively.

Table A.0.1: MilliAmpere parameters

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

Table A.0.2: Northern Clipper parameters

Parameter	Value
$l_{x,1}$	-35
$l_{y,1}$	7
$l_{x,2}$	-35
$l_{y,2}$	-7
$l_{x,3}$	35
m	$6000e^3$
L	76.2
g	9.81
$\boldsymbol{M}_{bis}$	$\begin{bmatrix} 1.1274 & 0 & 0 \\ 0 & 1.8902 & -0.0744 \\ 0 & -0.0744 & 0.1278 \end{bmatrix}$
$\boldsymbol{D}_{bis}$	$\begin{bmatrix} 0.0358 & 0 & 0 \\ 0 & 0.1183 & -0.0124 \\ 0 & -0.0041 & 0.0308 \end{bmatrix}$
$\boldsymbol{N}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & L \end{bmatrix}$

Table A.0.3: Objective function parameters

Parameter	Value
δ	10
$\boldsymbol{Q}$	Diag$([0, 200, 100])$
$\boldsymbol{R}$	Diag$([1e^{-2}, 1e^{-2}, 1e^{-2}, 1e^{-2}])$
$\boldsymbol{k}_s$	Diag$([1e^3, 1e^3, 1e^3, 1e^3, 1e^3])$
$\boldsymbol{Q}_\eta$	Diag$([10, 10, 1000])$
$\boldsymbol{Q}_\nu$	Diag$([0, 0, 0])$
$\boldsymbol{R}_f$	Diag$([1e^{-7}, 1e^{-7}, 1e^{-7}])$
$\boldsymbol{k}_c$	$[1e^3, 1e^3, 1e^3]^\top$
ρ	1
ϵ	$1e^{-3}$
$\boldsymbol{W}$	Diag$([1, 1, 1])$