Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2021

Modelling and Trajectory Planning for a Small-Scale Surface Ship

Fabian Steen and Gustav Zetterqvist

Master of Science Thesis in Electrical Engineering

Modelling and Trajectory Planning for a Small-Scale Surface Ship

Fabian Steen and Gustav Zetterqvist

LiTH-ISY-EX--21/5414--SE

Supervisor: **Fredrik Ljungberg**
ISY, Linköping University
Jonas Linder
ABB Corporate Research

Examiner: **Martin Enqvist**
ISY, Linköping University

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Autonomous ships are one way to increase safety at sea and to decrease environmental impact of marine traveling and shipping. For this application, a good representation of the environment and a physical model of the ship are vital components. By optimizing the trajectory of the ship, a good trade-off between the time duration and energy consumption can be found.

In this thesis, a three degree of freedom model that describes the dynamics of a small-scale surface ship is estimated. By using optimal control theory and a grey-box model, the parameters are estimated by defining an optimal control problem (OCP). The optimal solution is found by transcribing the problem into a nonlinear program and solving it using an interior point algorithm. The identification method is tested and validated using simulated data as well as using data from real world experiments. The performance of the estimated models is validated using cross validation.

In a second track of this thesis, a trajectory is created in two steps. The first is path planning to find a shortest geometric path between two points. In the second step, the path is converted to a trajectory and is optimized to become dynamically feasible. For this purpose, a roadmap is generated from a modified version of the generalized Voronoi diagram. To find an initial path in the roadmap, the A-star algorithm is utilized and to connect start and goal position to the map a few different methods are examined. An initial trajectory is created by mapping a straight-line trajectory to the initial path, thus connecting time, position and velocity. The final trajectory is found by solving a discrete OCP initialized with the initial trajectory. The OCP contains spatial constraints that ensures that the vessel does not collide with static obstacles.

The suggested estimation method resulted in models that could be used for trajectory planning to generate a dynamically feasible trajectory for both simulated and real data. The trajectory generated by the trajectory planner resulted in a collision-free trajectory, satisfying the dynamics of the estimated model, such that the trade-off between time duration and energy consumption is well balanced. Future work consists of implementation of a controller to see if the planned trajectory can be followed by the small-scale ship.

Acknowledgments

First off all, we could like to thank our supervisor Jonas Linder from ABB Corporate Research for his never-ending support and enthusiasm throughout this thesis work. He always took the time to answer our question and helped us move forward when we were stuck. We really appreciate all the time and effort he has put in to help us succeed with this thesis. Thanks to all parties at ABB trying their best to introduce us to our colleagues at ABB during these troubling times. Sadly, we could not come meet you personally, but thanks for all the laughs at the digital fika.

Thanks to Fredrik Ljungberg, our supervisor at Linköping University, for all your help and assistance. He was always available to answer any questions and gladly came by the lab to check on our progress. Thanks for all assistance and ideas regarding the model ship, without him it would probably not even float.

We would also like to express our gratitude to Martin Enqvist, our examiner, who suggested this thesis to us. He has provided valuable input and has always been positive and supportive.

Lastly, we would like to acknowledge our families and friends for their everlasting support and excitement during these years.

Linköping, June 2021
Fabian Steen & Gustav Zetterqvist

Contents

Notation

Notations

Notation	Description
$\boldsymbol{\eta}$	Position and orientation of ship
ψ	Heading of ship
$\boldsymbol{\nu}$	Linear and angular velocities of ship
$\boldsymbol{\tau}$	Forces and moments acting on ship
$\boldsymbol{\tau}_{\text{az}}$	Forces and moments from azimuth thrusters
$\boldsymbol{\tau}_t$	Forces and moments from tunnel thruster
$\boldsymbol{\tau}_r$	Forces and moments from rudders
$\boldsymbol{\tau}_w$	Forces and moments from wind
$\boldsymbol{R}(\psi)$	Rotation matrix
$\boldsymbol{M}$	Mass and inertia matrix
$\boldsymbol{C}(\boldsymbol{\nu})$	Coriolis matrix
$\boldsymbol{D}(\boldsymbol{\nu})$	Damping matrix
n_i	Propeller speed of thruster i
α_i	Angle of thruster i
$\boldsymbol{\vartheta}$	Model parameters

Reference Frames

Frame	Description
b-frame	Body-fixed reference frame, see Definition 2.1.
n-frame	Earth-fixed reference frame, see Definition 2.2.

Abbreviations

Abbreviation	Description
CG	Center of Gravity
CO	Center of Origin
CV	Cross Validation
DOF	Degrees Of Freedom
EKF	Extended Kalman Filter
EKS	Extended Kalman Smoother
GPS	Global Positioning System
GVD	Generalized Voronoi Diagram
IMU	Inertial Measurement Unit
IPOPT	Interior Point OPTimizer
MGVD	Modified Generalized Voronoi Diagram
NED	North-East-Down
NLP	Non-Linear Programming
NRMSE	Normalized Root Mean Square Error
OCP	Optimal Control Problem
RK4	4th order Runge-Kutta method
RTK	Real-Time Kinematic
RTS	Rauch–Tung–Striebel

1 Introduction

In this master's thesis, ways of estimating a mathematical model for a surface ship are explored, along with the problem of finding a feasible trajectory in a complex environment. The thesis work was performed at ABB Corporate Research in Västerås in collaboration with Linköping University.

1.1 Background

As technology advances, an increased number of tasks can be performed without human interaction. For example, autonomous cars and trucks are being tested in public, and with more than 80% of all world trade by volume being carried by sea (UNCTAD, 2020), autonomous ships are bound to follow. The main reasons for developing autonomous systems are to improve safety, reduce cost and decrease environmental impact (Wingrove, 2020).

Navigational errors, such as collision, contact and grounding/stranding, were the reason for 44% of all casualty events involving a ship with a connection to the EU between 2014-2019 (European Maritime Safety Agency, 2020). Out of all 1801 investigated accidents, 54% were attributed to human action. Futhermore, about 1000 personal injuries are reported every year of which 4% are very serious (European Maritime Safety Agency, 2020). With autonomous ships or automatic advisory systems many of these casualties could potentially be prevented and travelling by sea would be safer.

In marine shipping, the fuel cost stands for 57% of the total shipping cost (Furuichi and Shibasaki, 2015) and minimizing the fuel consumption would thus cut costs. One way to decrease fuel consumption is to have efficient path planning and control. This also leads to reduced impact on the environment with less

fuel emissions.

In the literature, autonomous ships are often categorized as autonomous surface vehicles (ASVs) or unmanned surface vehicles (USVs), these two terms are not to be confused with each other. An ASV is a ship that can operate and make decisions on its own, without human interaction, but is not necessarily unmanned, for example an ASV can transport passengers. An USV on the other hand does not need to be autonomous, and is often controlled by a human operator remotely (Vagale et al., 2020).

For the development of an ASV, many aspects need to be considered. The three most important aspects are the ship dynamics, navigation system and the control system (Peralta et al., 2020). In the navigation system one important aspect is motion planning (Peralta et al., 2020). Typically, a planning algorithm tries to find a series of actions that results in arriving at a specified goal state given an initial state (LaValle, 2006). For path planning, the initial and goal states are typically generalized positions, i.e. positions and orientations (LaValle, 2006).

The main purposes of path planning for ASVs are to

> [...] reduce the risk of collisions, groundings, and stranding accidents at sea, as well as costs and time expenditure. (Vagale et al., 2020)

For an ASV, the path planner plays a critical role in finding an optimal path to guide a vessel from one generalized position to another. It is desirable to optimize the path in some aspect while it should be dynamically feasible and collision-free. Some common criteria for optimization are to minimize path length, time duration and energy consumption (Vagale et al., 2020).

Path planning can be divided into local and global algorithms. The global path planning problem is to acquire a feasible path from the initial state to the goal state by assuming that a static map of the environment and information about obstacles are available. A local path planning algorithm considers information from sensors available on the ship to avoid dynamic obstacles in the vicinity of the vessel. Local path planning is sometimes called reactive path planning, due to the ability to react upon initially unknown obstacles. This causes the vessel to deviate from the original plan or adapt the speed if an obstacle intercepts its path (Vagale et al., 2020).

Often path planning is divided into multiple steps. First, an algorithm is applied to find a collision-free path between the initial state and the final state. This can be done either with or without respect to dynamics of the vessel. If the dynamics of the vessel is not considered, the path will not necessarily be dynamically feasible, thus the path may need to be refined. In the refinement algorithm, the path is transformed to follow all dynamic constraints and become feasible (LaValle, 2006). If the dynamic constraints are included in the collision-free path, a feasible path is achieved, but generally it is not optimal and it may be dangerously close to obstacles. Therefore, the refinement step aims at optimizing the collision-free path with some margin to get a faster or more efficient path (Bitar et al.,

2020).

Since a global path planning algorithm typically does not consider dynamical obstacles (Vagale et al., 2020), one way to avoid dynamic obstacles is to use a collision avoidance system in the controller. A control framework with the possibility to introduce such constraints is model predictive control (MPC). The MPC structure contains a goal function which is optimized under various constraints, e.g. dynamic constraints of the ship or obstacle constraints. This can be used to avoid collisions with dynamic obstacles simultaneously as controlling the ship to follow a given path (Eriksen and Breivik, 2017).

1.2 Problem Description

The focus of this master's thesis is modelling and path planning algorithms for autonomous ships. The objective is to find a dynamic model for the small-scale surface ship in Figure 1.1 and to develop a trajectory planning algorithm based on the model. The dynamic model is derived using a grey-box model based on physical principles, where the model parameters are estimated using system identification methods, i.e. using data collected from experiments.

1.3 Goals

The goal of this thesis is to determine a suitable grey-box model for ships, and then to estimate and validate the parameters of the small-scale ship using data collected from sea trials as well as selected experiments, such as a Bollard pull test. Further, a global path planning algorithm for ships is developed and later refined to fulfill the dynamic constraints of the small-scale ship.

1.4 Limitations

One limitation of this thesis is that all environmental disturbances, such as wind, waves and water currents are neglected to simplify both modelling and trajectory planning. In simulation, wind forces are included to investigate if the wind speed could be estimated from collected data. Futhermore, only the thruster configura-

Figure 1.1: *Hull of the model ship*

tion on the small-scale ship is considered in the modelling, system identification and the path planning algorithm. During data collection a global positioning system (GPS) and an inertial measurement unit (IMU) are used to measure the position and orientation of the ship. The GPS is assumed to have the possibility to use real-time kinematic (RTK) positioning to greatly increase the accuracy. In the path planning algorithm, a static map is used to generate a feasible path, and dynamic obstacles will be neglected.

1.5 Contributions

In this thesis, an optimal control based system identification method is implemented and tested with experimental data. The method is used to estimate the parameters of a 3 degrees of freedom (DOF) ship model with promising results. The identified model can be used for trajectory planning, control or simulation purposes. Furthermore, a trajectory planning algorithm is proposed based on the model of the ship. The generated trajectory combined with a controller can be used for navigation in confined waters to minimize the risk of collisions or grounding.

1.6 Outline

The thesis is outlined as follows. Chapter 2 includes the relevant theory on modelling of a ship and its thrusters, as well as descriptions of how to represent the forces created by the rudder and wind disturbances. Additionally, the proposed grey-box model of the ship is presented. Chapter 3 contains relevant theory in optimal control, used throughout the thesis. In Chapter 4, theory on model estimation and validation is given. Thereafter, the parameters of the grey-box model are estimated using both simulated data and data collected experimentally with the model ship. Chapter 5 begins with relevant theory and methodology for path planning. Then, different methods of how to connect start and goal positions to the path are presented. Chapter 5 concludes with a graph search method and methods to smoothen the obtained path. Thereafter, Chapter 6 presents theory on trajectory planning to achieve a feasible path for the model ship and obstacle avoidance. Lastly, in Chapter 7, conclusions and further work are presented.

2
Modelling

A mathematical model is useful to describe a physical system influenced by external forces. A model is also a powerful tool when designing advanced control systems and for path planning (Ljung and Glad, 2004).

In this chapter, a grey-box model for a ship is given. First, the dynamics of the ship are presented in Section 2.1 and then, Section 2.2 describes the external forces acting on the ship. Lastly, the full state-space model is presented in Section 2.3.

2.1 Ship Model

To fully describe a ships motion in all situations, 6DOF, surge, sway, heave, roll, pitch and yaw, should be considered. However, in calm seas where environmental disturbances, such as wind, waves and currents, are minimal, heave, roll and pitch are negligible (Fossen, 2011). A 3DOF model is therefore proposed to model the ship in this thesis. Using the notation from SNAME (1952) described in Table 2.1, the dynamics of the ship can be described by

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \quad \text{- Kinematics} \tag{2.1a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \quad \text{- Kinetics} \tag{2.1b}$$

where $\boldsymbol{R}(\psi)$ is a rotation matrix and $\boldsymbol{M}$, $\boldsymbol{C}(\boldsymbol{\nu})$ and $\boldsymbol{D}(\boldsymbol{\nu})$ are matrices describing the mass-inertia, Coriolis forces and damping of the ship, respectively (Fossen, 2011).

2.1.1 Reference Frames

A convenient way of describing the motion of a ship is by introducing reference frames, one body-fixed frame attached to the ship, denoted as the b-frame, and one Earth-fixed frame, called the n-frame (Fossen, 2011). These are defined according to Definition 2.1 and Definition 2.2, respectively. The relationship between the two reference frames can be seen in Figure 2.1(a).

Definition 2.1 (b-frame). The b–frame has its origin fixed to the ship's center of origin (CO), the x_b–axis facing towards the bow, the y_b–axis facing the starboard side and the z_b–axis facing downwards.

Definition 2.2 (n-frame). The n–frame has its origin fixed to the surface of the Earth, the x_n–axis facing North, the y_n–axis facing East and the z_n–axis facing the center of the Earth. This is also referred to as the North-East-down (NED) coordinate system.

2.1.2 Kinematics

The kinematics are described by the generalized position $\boldsymbol{\eta}$ and the generalized velocity $\boldsymbol{\nu}$. As can be seen in (2.1a), this geometric relation is

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

where $\boldsymbol{R}(\psi)$ is a rotation matrix given by

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

Table 2.1: *Notation used for describing the 3DOF motions of a ship (SNAME, 1952).*

Notation	Description
$\boldsymbol{\eta} = \begin{bmatrix} x & y & \psi \end{bmatrix}^T$	Generalized position in 3DOF given by Cartesian position (x, y) and heading angle ψ.
$\boldsymbol{\nu} = \begin{bmatrix} u & v & r \end{bmatrix}^T$	Generalized velocities in 3DOF given by surge velocity u, sway velocity v and yaw-rate r.
$\boldsymbol{\tau} = \begin{bmatrix} \tau_X & \tau_Y & \tau_N \end{bmatrix}^T$	Forces and moments affecting the ship in a body-fixed coordinate system in surge τ_X, sway τ_Y and yaw τ_N.

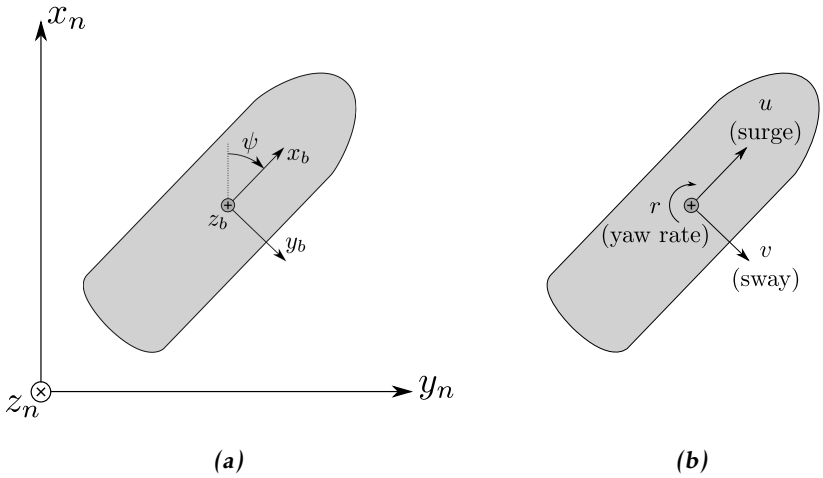


Figure 2.1: *Reference frames for a ship. (a) illustrates the relationship between the b-frame and the n-frame, where ψ is the heading of the ship. In (b) the generalized velocities in the b-frame are shown.*

2.1.3 Kinetics

The kinetics describe the motion of the ship under the influence of external forces. The kinetics are described by the generalized velocity $\boldsymbol{\nu}$ and the generalized external forces $\boldsymbol{\tau}$, both expressed in the b-frame. From (2.1b), the relationship is

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

where $\boldsymbol{M}$, $\boldsymbol{C}(\boldsymbol{\nu})$ and $\boldsymbol{D}(\boldsymbol{\nu})$ are matrices describing the mass-inertia, Coriolis forces and damping of the ship, respectively (Fossen, 2011). The generalized forces acting on the ship $\boldsymbol{\tau}$ are described separately in Section 2.2.

As a ship moves through water, the surrounding water is displaced and set into motion. It is common to separate the mass and inertia matrix $\boldsymbol{M}$, along with the Coriolis matrix $\boldsymbol{C}(\boldsymbol{\nu})$, into one part for the rigid-body kinetics and one part for the added mass due to hydrodynamic effects, such that

$$\boldsymbol{M} = \boldsymbol{M}_{\text{RB}} + \boldsymbol{M}_A \tag{2.3a}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{\text{RB}}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}). \tag{2.3b}$$

With the b-frame defined as in Definition 2.1, and assuming that the ship is symmetric in the x_b-z_b-plane, the surge motion can be decoupled from the sway- and yaw motion (Fossen, 2011). Furthermore, by assuming that the center of gravity

(CG) coincides with CO, the mass and inertia matrix $\boldsymbol{M}$ is given by

$$\boldsymbol{M} \coloneqq \underbrace{\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix}}_{\boldsymbol{M}_{\mathrm{RB}}} + \underbrace{\begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & 0 \\ 0 & 0 & -N_{\dot{r}} \end{bmatrix}}_{\boldsymbol{M}_A} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix}, \tag{2.4}$$

where m is the mass, I_z is the moment of inertia about the z_b-axis and the parameters $X_{\dot{u}}$, $Y_{\dot{v}}$ and $N_{\dot{r}}$ are referred to as hydrodynamic derivatives. From a control perspective, only the sum of mass-inertia and hydrodynamical parameters are important. Also, when identifying the parameters experimentally it is difficult to separate them, thus the terms are added together for simplification purposes.

The Coriolis matrix can be described as a function of the mass and inertia matrix $\boldsymbol{M}$ and the generalized velocities $\boldsymbol{\nu}$

$$\boldsymbol{C}(\boldsymbol{\nu}) \coloneqq \underbrace{\begin{bmatrix} 0 & 0 & -mv \\ 0 & 0 & mu \\ mv & -mu & 0 \end{bmatrix}}_{\boldsymbol{C}_{\mathrm{RB}}(\boldsymbol{\nu})} + \underbrace{\begin{bmatrix} 0 & 0 & Y_{\dot{v}} v \\ 0 & 0 & -X_{\dot{u}} u \\ -Y_{\dot{v}} v & X_{\dot{u}} u & 0 \end{bmatrix}}_{\boldsymbol{C}_A(\boldsymbol{\nu})} = \begin{bmatrix} 0 & 0 & -m_{22} v \\ 0 & 0 & m_{11} u \\ m_{22} v & -m_{11} u & 0 \end{bmatrix}, \tag{2.5}$$

where the last equality follows from (2.4).

The hydrodynamic damping consists of a linear part and a non-linear part, but it is convenient to combine them to get the damping matrix

$$\boldsymbol{D}(\boldsymbol{\nu}) \coloneqq \begin{bmatrix} d_{11}(\boldsymbol{\nu}) & 0 & 0 \\ 0 & d_{22}(\boldsymbol{\nu}) & d_{23}(\boldsymbol{\nu}) \\ 0 & d_{32}(\boldsymbol{\nu}) & d_{33}(\boldsymbol{\nu}) \end{bmatrix}, \tag{2.6}$$

with

$$d_{11}(\boldsymbol{\nu}) = X_u - X_{|u|u}|u| \tag{2.7a}$$
$$d_{22}(\boldsymbol{\nu}) = Y_v - Y_{|u|v}|u| - Y_{|v|v}|v| - Y_{|r|v}|r| \tag{2.7b}$$
$$d_{23}(\boldsymbol{\nu}) = Y_r \tag{2.7c}$$
$$d_{32}(\boldsymbol{\nu}) = N_v - N_{|v|v}|v| \tag{2.7d}$$
$$d_{33}(\boldsymbol{\nu}) = N_r - N_{|r|r}|r|, \tag{2.7e}$$

where the parameters will be identified experimentally. Additional non-linear terms can be added, but those mentioned in (2.7) are assumed to suffice for these applications (Teeuwen, 2018).

To ensure the stability of the ship, (2.1b) is linearized at constant surge velocity

$(u = U_0, v = r = 0)$, which yields

$$\begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix} \dot{\boldsymbol{\nu}} + \underbrace{\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v - Y_{|u|v}|U_0| & Y_r + m_{11}U_0 \\ 0 & N_v - (m_{11} - m_{22})U_0 & N_r \end{bmatrix}}_{:=\boldsymbol{N}} \boldsymbol{\nu} = \boldsymbol{\tau}, \tag{2.8}$$

where, in addition, $X_{|u|u}$ has been neglected for simplicity. In Mucha (2017), it is proven that for straight line stability, i.e., when moving in a straight line, the matrix $\boldsymbol{N}$ needs to be positive definite, which yields

$$X_u > 0 \tag{2.9a}$$

$$Y_v > 0 \tag{2.9b}$$

$$N_r > 0 \tag{2.9c}$$

$$\left(Y_v - Y_{|u|v}|U_0|\right) N_r - \left(m_{11}U_0 + Y_r\right)\left(N_v - \left(m_{11} - m_{22}\right)U_0\right) > 0. \tag{2.9d}$$

If (2.9d) is fulfilled for $U_0 = 0$ as well as for the maximum surge velocity the ship can achieve ($U_0 = U_{\max}$), stability for all velocities $0 \leq u \leq U_{\max}$ can be shown to be guaranteed if $m_{11} < m_{22}$.

2.2 External Forces

There are a lot of forces acting on a ship, in this section, the most prominent of these will be addressed. Firstly, actuators on the ship, such as propellers and rudders, cause a controllable force on the ship, called $\boldsymbol{\tau}_c$. Secondly, there are forces caused by environmental changes in the vicinity of the ship, denoted by $\boldsymbol{\tau}_{\text{env}}$. Thus, the combined generalized force affecting the ship is assumed to be

$$\boldsymbol{\tau} = \boldsymbol{\tau}_c + \boldsymbol{\tau}_{\text{env}}. \tag{2.10}$$

With the thruster configuration of the investigated ship, as seen in Figure 2.2, the controllable generalized force is assumed to be divided into a propulsion force from the azimuth thrusters $\boldsymbol{\tau}_{\text{az}}$ and the tunnel thruster $\boldsymbol{\tau}_t$, as well as a rudder force $\boldsymbol{\tau}_r$ caused by the duct of the azimuth thrusters. The total controllable generalized force is thereby

$$\boldsymbol{\tau}_c = \boldsymbol{\tau}_{\text{az}} + \boldsymbol{\tau}_t + \boldsymbol{\tau}_r. \tag{2.11}$$

The environmental forces include forces from wind, currents and waves. For simplicity, only disturbances from the wind will be considered in this thesis, thus

$$\boldsymbol{\tau}_{\text{env}} \approx \boldsymbol{\tau}_w, \tag{2.12}$$

where $\boldsymbol{\tau}_w$ is the generalized force generated by the wind.

2.2.1 Azimuth Thrusters

On the ship there are two azimuth thrusters located in the aft. They can be rotated about their z_b-axis to create a force in any direction in the x_b-y_b-plane.

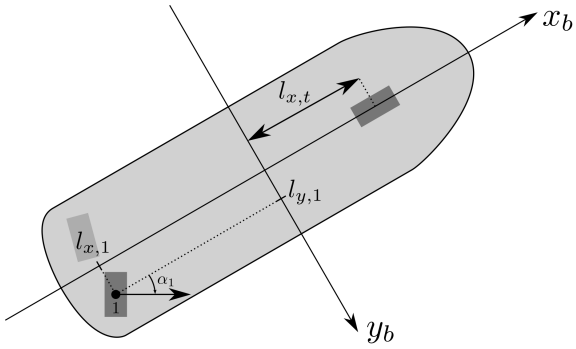


Figure 2.2: *An overview of the ship with thruster locations, where $l_{x,1}$ and $l_{y,1}$ are the position of thruster 1 in the b-frame and α_1 is the angle of thruster 1. The position of the tunnel thruster on the x_b-axis is denoted by $l_{x,t}$. The parameters for thruster 2 is similar to thruster 1, but they are left out for readability.*

Let α_i denote the angle and n_i denote the propeller angular velocity of thruster i, where $i = 1, 2$, are the indices of the thrusters. The angle α_i is assumed to be zero when thruster i is pointing towards the bow of the ship and increases with rotation about the z_b-axis. Assuming that both thrusters are equally efficient, the generalized force from the azimuth thrusters can be described with

$$\boldsymbol{\tau}_{\text{az}} = \begin{bmatrix} (\mu_{\text{az}} - \kappa u)\, \tilde{\tau}_X \\ \mu_{\text{az}} \tilde{\tau}_Y \\ \mu_{\text{az}} \tilde{\tau}_N \end{bmatrix}, \tag{2.13}$$

where μ_{az} and κ are positive constants and

$$\tilde{\boldsymbol{\tau}} = \begin{bmatrix} \tilde{\tau}_X \\ \tilde{\tau}_Y \\ \tilde{\tau}_N \end{bmatrix} := \begin{bmatrix} \sum_{i=1}^{2} n_i^2 \cos(\alpha_i) \\ \sum_{i=1}^{2} n_i^2 \sin(\alpha_i) \\ \sum_{i=1}^{2} n_i^2 \left(l_{x,i} \sin(\alpha_i) - l_{y,i} \cos(\alpha_i) \right) \end{bmatrix}, \tag{2.14}$$

where $l_{x,i}$ and $l_{y,i}$ describes the mounting position in the x_b-y_b-plane of thruster i in the b-frame as seen in Figure 2.2 (Lewandowski, 2004). Note that $\tilde{\boldsymbol{\tau}}$ is known if the system inputs, α_i and n_i, as well as the mounting positions of the thrusters are known.

2.2.2 Tunnel Thruster

The tunnel thruster is a transversal propeller built into the hull of the ship and can be modelled as a fixed azimuth thruster with $\alpha_t = 90°$. Assuming the tunnel thruster is mounted on the center line of the ship, the generalized force from the

tunnel thruster is

$$\boldsymbol{\tau}_t = \mu_t n_t^2 \begin{bmatrix} 0 \\ 1 \\ l_{x,t} \end{bmatrix}, \tag{2.15}$$

where μ_t is a positive constant and n_t is the propeller speed of the tunnel thruster. The position on the x_b-axis is denoted by $l_{x,t}$.

2.2.3 Rudder Model

The configuration of the ship has no rudder, however, the duct of the azimuth thrusters potentially cause a force comparable with a rudder force. The generalized force generated from a rudder can be described by

$$\boldsymbol{F}_r(\delta, \boldsymbol{\nu}_r) = \begin{bmatrix} -F_N(\delta, \boldsymbol{\nu}_r)\sin(\delta) \\ F_N(\delta, \boldsymbol{\nu}_r)\cos(\delta) \\ F_N(\delta, \boldsymbol{\nu}_r)\left(l_{x,r}\cos(\delta) + l_{y,r}\sin(\delta)\right) \end{bmatrix}, \tag{2.16}$$

where δ is the rudder angle, $l_{x,r}$ and $l_{y,r}$ describe the mounting position of the rudder in the b-frame and

$$F_N(\delta, \boldsymbol{\nu}_r) = \frac{1}{2}\rho_w A_r F'_N(\delta, \boldsymbol{\nu}_r)\left(u_r^2 + v_r^2\right) \tag{2.17a}$$

$$F'_N(\delta, \boldsymbol{\nu}_r) = C_N \sin(\delta - \operatorname{atan2}(v_r, u_r)), \tag{2.17b}$$

where ρ_w is the water density, A_r is the effective rudder area, C_N is a positive constant, u_r and v_r are the surge and sway velocities at the rudder, respectively (Kang et al., 2008). In Kang et al. (2008), experiments found that $F'_N(\delta, \boldsymbol{\nu}_r)$ is not perfectly modelled as in (2.17b), especially at high rudder angles where stall occurs. Nevertheless, this approximation will suffice for this thesis. Thus, the total generalized force generated by rudder effects from the two azimuth thrusters is

$$\boldsymbol{\tau}_r = \begin{bmatrix} \sum_{i=1}^{2} -F_N(\alpha_i, \boldsymbol{\nu})\sin(\alpha_i) \\ \sum_{i=1}^{2} F_N(\alpha_i, \boldsymbol{\nu})\cos(\alpha_i) \\ \sum_{i=1}^{2} F_N(\alpha_i, \boldsymbol{\nu})\left(l_{x,i}\cos(\alpha_i) + l_{y,i}\sin(\alpha_i)\right) \end{bmatrix}, \tag{2.18}$$

where the flow velocities at the rudder are approximated with the velocities of the ship. In the rudder model, C_N is the only parameter to be estimated.

2.2.4 Wind Model

To consider forces from the wind acting on the ship a simple wind model proposed by Fossen (2011) is

$$\boldsymbol{\tau}_w = \frac{1}{2}\rho_a V_{\text{rw}}^2 \begin{bmatrix} C_X(\gamma_{\text{rw}})A_{\text{fw}} \\ C_Y(\gamma_{\text{rw}})A_{\text{lw}} \\ C_N(\gamma_{\text{rw}})A_{\text{lw}}L_{\text{oa}} \end{bmatrix}, \tag{2.19}$$

where ρ_a is the density of air, L_{oa} is the length overall of the ship, A_{fw} and A_{lw} are the frontal and lateral projected areas of the ship, respectively. The length

overall L_{oa} is the maximum length of the ship that is parallel to the waterline. Furthermore,

$$V_{\text{rw}} = \sqrt{u_{\text{rw}}^2 + v_{\text{rw}}^2} \tag{2.20a}$$

$$C_X(\gamma_{\text{rw}}) = -c_x \cos(\gamma_{\text{rw}}) \tag{2.20b}$$

$$C_Y(\gamma_{\text{rw}}) = c_y \sin(\gamma_{\text{rw}}) \tag{2.20c}$$

$$C_N(\gamma_{\text{rw}}) = c_n \sin(2\gamma_{\text{rw}}) \tag{2.20d}$$

$$\gamma_{\text{rw}} = -\text{atan2}(v_{\text{rw}}, u_{\text{rw}}), \tag{2.20e}$$

where u_{rw} and v_{rw} are the relative surge and sway velocities in the b-frame, respectively, and c_x, c_y and c_n are the wind coefficients for the ship and in Fossen (2011) it is stated that $c_x \in [0.50, 0.90]$, $c_x \in [0.70, 0.95]$ and $c_x \in [0.05, 0.20]$. The relative velocities can be described by

$$u_{\text{rw}} = u - u_w \tag{2.21a}$$

$$v_{\text{rw}} = v - v_w \tag{2.21b}$$

$$u_w = V_w \cos(\beta_w - \psi) \tag{2.21c}$$

$$v_w = V_w \sin(\beta_w - \psi), \tag{2.21d}$$

where u_w and v_w are the wind velocities in the b-frame, V_w is the speed of the wind and β_w is the wind direction in the n-frame, calculated as the angle starting from the positive y_n-axis and increasing by rotation about the positive z_n-axis (Fossen, 2011).

2.3 State-Space Model

To identify the unknown parameters experimentally, the inputs and outputs of the system need to be measured. The inputs are propeller speed and thruster angle of all thrusters, and the outputs are Cartesian coordinates in the n-frame, the heading angle as well as yaw-rate of the ship. A standard notation of a state-space model is

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\vartheta}) \tag{2.22a}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\vartheta}), \tag{2.22b}$$

where $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ is the input vector, $\boldsymbol{\vartheta}$ is a constant vector containing the model parameters and $\boldsymbol{y}$ is the output vector.

Rewriting (2.1b) on this form yields

$$\begin{aligned}\dot{\boldsymbol{\nu}} &= \begin{bmatrix}\dot{u} & \dot{v} & \dot{r}\end{bmatrix}^T = \boldsymbol{M}^{-1}\left(-\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{\tau}\right) \\ &= \begin{bmatrix}\frac{1}{m_{11}} & 0 & 0 \\ 0 & \frac{1}{m_{22}} & 0 \\ 0 & 0 & \frac{1}{m_{33}}\end{bmatrix}\left(\begin{bmatrix} m_{22}vr - X_u u + X_{|u|u}|u|u \\ -m_{11}ur - Y_v v - Y_r r + Y_{|u|v}|u|v + Y_{|v|v}|v|v + Y_{|r|v}|r|v \\ (m_{11} - m_{22})uv - N_v v + N_{|v|v}|v|v - N_r r + N_{|r|r}|r|r \end{bmatrix}\right. \\ &\quad \left. + \begin{bmatrix}\tau_X(\boldsymbol{u}, \boldsymbol{\vartheta}) \\ \tau_Y(\boldsymbol{u}, \boldsymbol{\vartheta}) \\ \tau_N(\boldsymbol{u}, \boldsymbol{\vartheta})\end{bmatrix}\right) \\ &= \begin{bmatrix} \frac{1}{m_{11}}\left(m_{22}vr - X_u u + X_{|u|u}|u|u + \tau_X(\boldsymbol{u}, \boldsymbol{\vartheta})\right) \\ \frac{1}{m_{22}}\left(-m_{11}ur - Y_v v - Y_r r + + Y_{|u|v}|u|v + Y_{|v|v}|v|v + Y_{|r|v}|r|v + \tau_Y(\boldsymbol{u}, \boldsymbol{\vartheta})\right) \\ \frac{1}{m_{33}}\left((m_{11} - m_{22})uv - N_v v + N_{|v|v}|v|v - N_r r + N_{|r|r}|r|r + \tau_N(\boldsymbol{u}, \boldsymbol{\vartheta})\right)\end{bmatrix}. \end{aligned} \tag{2.23}$$

By including (2.1a) and the measurements, the full state-space model is given by

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \cos(\psi)u - \sin(\psi)v \\ \sin(\psi)u + \cos(\psi)v \\ r \\ \frac{1}{m_{11}}\left(m_{22}vr - X_u u + X_{|u|u}|u|u + \tau_X(\boldsymbol{u}, \boldsymbol{\vartheta})\right) \\ \frac{1}{m_{22}}\left(-m_{11}ur - Y_v v - Y_r r + + Y_{|u|v}|u|v + Y_{|v|v}|v|v + Y_{|r|v}|r|v + \tau_Y(\boldsymbol{u}, \boldsymbol{\vartheta})\right) \\ \frac{1}{m_{33}}\left((m_{11} - m_{22})uv - N_v v + N_{|v|v}|v|v - N_r r + N_{|r|r}|r|r + \tau_N(\boldsymbol{u}, \boldsymbol{\vartheta})\right)\end{bmatrix}}_{:= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\vartheta})} \tag{2.24a}$$

$$\boldsymbol{y} = \begin{bmatrix} x \\ y \\ \psi \\ r \end{bmatrix} := \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\vartheta}), \tag{2.24b}$$

where $\boldsymbol{x} = \begin{bmatrix} x & y & \psi & u & v & r \end{bmatrix}^T$ is the state, $\boldsymbol{u} = \begin{bmatrix} \alpha_1 & \alpha_2 & n_1 & n_2 & n_t \end{bmatrix}^T$ is the input, $\boldsymbol{y}$ is the output and

$$\boldsymbol{\vartheta} = \begin{bmatrix} m_{11} & m_{22} & m_{33} & X_u & X_{|u|u} & Y_v & Y_r & Y_{|u|v} & Y_{|v|v} & Y_{|r|v} & N_v & N_r & N_{|v|v} & N_{|r|r} & \kappa & C_N & V_w & \beta_w & c_x & c_y & c_n & \mu_{\text{az}} & \mu_t \end{bmatrix}^T \tag{2.25}$$

is the vector of unknown parameters. The external forces are

$$\boldsymbol{\tau} = \begin{bmatrix}\tau_X(\boldsymbol{u}, \boldsymbol{\vartheta}) \\ \tau_Y(\boldsymbol{u}, \boldsymbol{\vartheta}) \\ \tau_N(\boldsymbol{u}, \boldsymbol{\vartheta})\end{bmatrix} \tag{2.26}$$

and they are given in Section 2.2.

3

Optimal Control

Optimal control is a useful tool when dealing with system identification as well as optimization of trajectories. In this chapter, the theory on optimal control is presented.

In Section 3.1 a general optimal control problem is formulated, and in Section 3.2 a numerical method to solve the problem is presented.

3.1 Optimal Control Problem

For an optimal control problem (OCP), the aim is to minimize a cost function given the system dynamics, a control variable and boundary constraints. In general, the cost function is given by

$$\phi\left(\boldsymbol{x}(t_f)\right) + \int_{t_i}^{t_f} f_0\left(t, \boldsymbol{x}, \boldsymbol{u}\right) dt \tag{3.1}$$

where the first term penalizes the deviation from some desired final state and the integral part is a cost associated with the state and control trajectories. The system dynamics is defined in terms of a state space equation $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\vartheta})$, the control variable is restricted to $\boldsymbol{u} \in \mathcal{U}$ and the state vector $\boldsymbol{x}$ is constrained to $\boldsymbol{x} \in \mathcal{S}$. The initial and final times are denoted by t_i and t_f, respectively. This

results in the optimal control problem

$$\min_{\boldsymbol{u}} \ \phi\left(\boldsymbol{x}(t_f)\right) + \int_{t_i}^{t_f} f_0\left(t, \boldsymbol{x}, \boldsymbol{u}\right) dt \tag{3.2a}$$

$$\text{s.t. } \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\vartheta}) \tag{3.2b}$$

$$\boldsymbol{x} \in \mathcal{S} \tag{3.2c}$$

$$\boldsymbol{u} \in \mathcal{U}. \tag{3.2d}$$

The optimal control problem is thus to find an admissible control signal $\boldsymbol{u} \in \mathcal{U}$, such that the state stays within the set $\mathcal{S}$ with minimal cost (Jönsson et al., 2010).

3.2 Multiple Shooting

To solve the OCP, a numerical approach called multiple shooting can be applied. The multiple shooting algorithm is described in Algorithm 3.1, and results in a nonlinear programming (NLP) problem on the form

$$\min_{\boldsymbol{w}} \ \Phi(\boldsymbol{w}) \tag{3.3a}$$

$$\text{s.t. } \boldsymbol{G}(\boldsymbol{w}) = 0 \tag{3.3b}$$

$$\boldsymbol{H}(\boldsymbol{w}) \leq 0, \tag{3.3c}$$

where $\Phi(\boldsymbol{w})$ is the objective function, $\boldsymbol{w} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_{N+1}^T, \boldsymbol{u}_1^T, \boldsymbol{u}_2^T, \ldots, \boldsymbol{u}_N^T, \boldsymbol{\vartheta}^T]^T$ is the decision vector containing the decision variables where the states and control signals are discretized. Furthermore, the equality constraints $\boldsymbol{G}(\boldsymbol{w})$ includes the shooting constraints and the inequality constraints (3.3c) corresponds to the constraints on the state, control signal and parameter vector.

For the shooting constraints, a discrete-time model $\boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is defined using the 4th order Runge-Kutta (RK4) method

$$\boldsymbol{k}_1 = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{3.4a}$$

$$\boldsymbol{k}_2 = \boldsymbol{f}(\boldsymbol{x}_k + \frac{h}{2}\boldsymbol{k}_1, \boldsymbol{u}_k) \tag{3.4b}$$

$$\boldsymbol{k}_3 = \boldsymbol{f}(\boldsymbol{x}_k + \frac{h}{2}\boldsymbol{k}_2, \boldsymbol{u}_k) \tag{3.4c}$$

$$\boldsymbol{k}_4 = \boldsymbol{f}(\boldsymbol{x}_k + h\,\boldsymbol{k}_3, \boldsymbol{u}_k) \tag{3.4d}$$

$$\boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{x}_k + \frac{h}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right), \tag{3.4e}$$

where h is the discretization time step. Given a state $\boldsymbol{x}_k$ and a control input $\boldsymbol{u}_k$, the next iteration can be defined as $\boldsymbol{x}_{k+1} = \boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{u}_k)$, thus the equality constraints

becomes

$$\boldsymbol{G}(\boldsymbol{w}) = \begin{bmatrix} \boldsymbol{F}(\boldsymbol{x}_1, \boldsymbol{u}_1) - \boldsymbol{x}_2 \\ \boldsymbol{F}(\boldsymbol{x}_2, \boldsymbol{u}_2) - \boldsymbol{x}_3 \\ \dots \\ \boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N) - \boldsymbol{x}_{N+1} \\ \boldsymbol{g}(\boldsymbol{w}) \end{bmatrix}, \tag{3.5}$$

where $\boldsymbol{g}(\boldsymbol{w})$ contains the equality constraints other than the shooting constraints.

Algorithm 3.1: Direct multiple shooting (Bock and Plitt, 1984).

1 Discretize the time horizon, $[t_i, t_f]$, into subintervals $[t_i, t_{i+1}]$, such that

$$t_i = t_1 < t_2 < \dots < t_{N+1} = t_f, \tag{3.6}$$

where N is the number of subintervals.

2 Approximate the control signal as piecewise constant on all subintervals

$$\boldsymbol{u}(t) = \boldsymbol{v}_i \text{ for } t \in [t_i, t_{i+1}] \text{ and } i = 1, 2, \dots, N. \tag{3.7}$$

3 Choose the initial values of the states at each time instance

$$\boldsymbol{x}(t_i) = \boldsymbol{h}_i \text{ for } i = 1, 2, \dots, N. \tag{3.8}$$

4 Calculate the state trajectories in each subinterval using the initial value and the system dynamics

$$\dot{\boldsymbol{x}}_i = \boldsymbol{f}(\boldsymbol{x}_i(t), \boldsymbol{v}_i(t), \boldsymbol{\vartheta}), t \in [t_i, t_{i+1}] \tag{3.9a}$$

$$\boldsymbol{x}_i(t_i) = \boldsymbol{h}_i \text{ for } i = 1, 2, \dots, N. \tag{3.9b}$$

5 Ensure continuity of the solution by adding the shooting constraints

$$\boldsymbol{h}_{i+1} - \boldsymbol{x}_i(t_{i+1}; \boldsymbol{h}_i, \boldsymbol{v}_i) = 0 \text{ for } i = 1, 2, \dots, N. \tag{3.10}$$

6 Compute the objective function for each subinterval, then solve the NLP

$$\min_{\boldsymbol{v}_i} \ \phi(\boldsymbol{x}_{N+1}) + \sum_{i=1}^{N} \int_{t_i}^{t_{i+1}} f_0(\boldsymbol{x}_i(t), \boldsymbol{v}_i) dt \tag{3.11a}$$

$$\text{s.t. } \boldsymbol{h}_{i+1} - \boldsymbol{x}_i(t_{i+1}; \boldsymbol{h}_i, \boldsymbol{v}_i) = 0, \ i = 1, 2, \dots, N \tag{3.11b}$$

$$\boldsymbol{x}_i \in \mathcal{S} \tag{3.11c}$$

$$\boldsymbol{v}_i \in \mathcal{U}. \tag{3.11d}$$

4

Model Estimation

For the estimation of the unknown parameters in the grey-box model, an optimal control problem is formed and solved using a multiple shooting method. The OCP is equivalent to a non-linear least square error problem using an non-linear output error-model.

This chapter threats the theory and results for the estimation of the grey-box model derived in Chapter 2. The chapter begins with Section 4.1, where it is explained how the data was gathered. Then, the theory on how to estimate the parameters is presented in Section 4.2. A method is thereafter proposed to validate an estimated model in Section 4.3. In Section 4.4 the experimental platform is described. Lastly, the results from the simulated and experimental model estimation are presented in Section 4.5

4.1 Data Collection

During the experiments, described in Appendix A, data was collected in datasets. Each experiment has a separate dataset of N_i data points, denoted by

$$\mathcal{D}_i(N_i) = \left\{\bar{\boldsymbol{y}}_i(k), \bar{\boldsymbol{u}}_i(k)\right\}_{k=1}^{N_i}, \tag{4.1}$$

where i is the experiment index and $\bar{\boldsymbol{y}}$ and $\bar{\boldsymbol{u}}$ represents the measurement of $\boldsymbol{y}$ and $\boldsymbol{u}$, respectively. The datasets are then divided into estimation data and validation data, where the former will be used for estimating the model and the latter will be used for validation of the estimated model. All the estimation and validation data will then be gathered into two datasets, denoted by $\mathcal{D}_e(N_e)$ and $\mathcal{D}_v(N_v)$, respectively, where $N = N_e \approx N_v$ is the number of data points in each dataset.

4.2 Parameter Estimation

With a chosen model structure, the parameters $\boldsymbol{\vartheta}$, can be estimated by solving

$$\hat{\boldsymbol{\vartheta}}_N = \underset{\boldsymbol{\vartheta}}{\operatorname{argmin}} V_N(\boldsymbol{\vartheta}, \mathcal{D}_e(N)), \tag{4.2}$$

where $V_N(\cdot)$ is a value function depending on the model structure and the estimation data, and N denotes the number of data points used for the optimization problem. The value function, $V_N(\cdot)$, can have different structures, and each structure will correspond to a unique estimator. A commonly used estimator is the weighted least squares (WLS), which is defined by using the value function

$$V_N^{WLS}(\boldsymbol{\vartheta}, \mathcal{D}_e(N)) = \sum_{k=1}^{N} \left(\bar{\boldsymbol{y}}(k) - \hat{\boldsymbol{y}}(k|\boldsymbol{\vartheta})\right)^T \boldsymbol{W} \left(\bar{\boldsymbol{y}}(k) - \hat{\boldsymbol{y}}(k|\boldsymbol{\vartheta})\right), \tag{4.3}$$

where $\boldsymbol{W}$ is a weighing matrix, $\bar{\boldsymbol{y}}$ is the measured output and $\hat{\boldsymbol{y}}$ is the predicted output given $\boldsymbol{\vartheta}$ (Gustafsson, 2018).

To be able to distinguish between different models, the dataset has to be informative enough, otherwise the problem will not converge to the global minimum.

4.2.1 Problem Formulation

The optimization problem (4.2) can be seen as an OCP, with the notation used in Section 2.3, it can be expressed as

$$\min_{\boldsymbol{x},\boldsymbol{\vartheta}} \int_0^{t_f} L\left(\hat{\boldsymbol{y}}(t|\boldsymbol{\vartheta}), \bar{\boldsymbol{y}}(t)\right) dt \tag{4.4a}$$

$$\text{s.t. } \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\vartheta}) \tag{4.4b}$$

$$\boldsymbol{g}(\boldsymbol{\vartheta}) \leq 0, \tag{4.4c}$$

where (4.4a) corresponds to the optimization problem (4.2), (4.4b) are the dynamic equations of the ship (2.24a) and (4.4c) are the inequality constraints on the parameters (2.25). Since the dynamics in (4.4b) are nonlinear, the problem is non-convex, meaning that the problem might have many local minima (Boyd and Vandenberghe, 2004). As a consequence, a good initial guess is often necessary for the problem to converge to the global minimum (Boyd and Vandenberghe, 2004).

To solve the OCP (4.4), multiple shooting is applied, this results in a NLP on the form

$$\min_{\boldsymbol{w}} \ \Phi(\boldsymbol{w}, \bar{\boldsymbol{y}}) \tag{4.5a}$$

$$\text{s.t. } \boldsymbol{G}(\boldsymbol{w}, \bar{\boldsymbol{u}}) = 0 \tag{4.5b}$$

$$\boldsymbol{H}(\boldsymbol{w}) \leq 0, \tag{4.5c}$$

with

$$\Phi(\boldsymbol{w}, \bar{\boldsymbol{y}}) = \sum_{i=1}^{N} \left(\bar{\boldsymbol{y}}_i - \boldsymbol{y}_i\right)^T \boldsymbol{W} \left(\bar{\boldsymbol{y}}_i - \boldsymbol{y}_i\right) \tag{4.6}$$

where $\boldsymbol{w} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_{N+1}^T, \boldsymbol{\vartheta}^T]^T$ is the decision vector containing the discrete system state $\boldsymbol{x}_k$ and the model parameters $\boldsymbol{\vartheta}$. The measured output is denoted by $\bar{\boldsymbol{y}} = [\bar{\boldsymbol{y}}_1^T, \bar{\boldsymbol{y}}_2^T, \ldots, \bar{\boldsymbol{y}}_N^T]^T$, while $\boldsymbol{y}_k$ denotes the output given the system state $\boldsymbol{x}_k$. Further, the shooting constraints is $\boldsymbol{G}(\boldsymbol{w}, \bar{\boldsymbol{u}})$ where $\bar{\boldsymbol{u}} = [\bar{\boldsymbol{u}}_1^T, \bar{\boldsymbol{u}}_2^T, \ldots, \bar{\boldsymbol{u}}_N^T]^T$ is the measured control input. Lastly, the inequality constraints (3.3c) corresponds to the constraints on the model parameters $\boldsymbol{\vartheta}$.

For the shooting constraints, a discrete-time model $\boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is defined using the RK4 method, which yields

$$\boldsymbol{G}(\boldsymbol{w}, \bar{\boldsymbol{u}}) = \begin{bmatrix} \boldsymbol{F}(\boldsymbol{x}_1, \bar{\boldsymbol{u}}_1) - \boldsymbol{x}_2 \\ \boldsymbol{F}(\boldsymbol{x}_2, \bar{\boldsymbol{u}}_2) - \boldsymbol{x}_3 \\ \cdots \\ \boldsymbol{F}(\boldsymbol{x}_N, \bar{\boldsymbol{u}}_N) - \boldsymbol{x}_{N+1} \end{bmatrix}. \tag{4.7}$$

The constraints on the parameters from Chapter 2 can be summarized as

$$X_u > 0 \tag{4.8a}$$
$$Y_v > 0 \tag{4.8b}$$
$$N_r > 0 \tag{4.8c}$$
$$\left(Y_v - Y_{|u|v}|U_0|\right) N_r - (m_{11} U_0 + Y_r)(N_v - (m_{11} - m_{22}) U_0) > 0 \tag{4.8d}$$
$$Y_v N_r - Y_r N_v > 0 \tag{4.8e}$$
$$m_{22} - m_{11} > 0 \tag{4.8f}$$
$$\kappa > 0 \tag{4.8g}$$
$$0.50 \leq c_X \leq 0.90 \tag{4.8h}$$
$$0.70 \leq c_Y \leq 0.95 \tag{4.8i}$$
$$0.05 \leq c_N \leq 0.20. \tag{4.8j}$$

4.2.2 Extended Kalman Filter

For the Multiple shooting algorithm to converge, a good initial guess of the decision vector $\boldsymbol{w}$ is crucial. Therefore, an extended Kalman filter (EKF) is applied to estimate $\boldsymbol{\nu}$ from measurements of $\boldsymbol{\eta}$. This algorithm consists of a prediction step, where the states of the system are predicted using a constant velocity state-space model, and a measurement update step, where the measurements are used to correct the states (Gustafsson, 2018). Optionally, a Rauch–Tung–Striebel (RTS) smoother can be applied afterwards to smoothen the estimates (Gustafsson, 2018), this is referred to as an extended Kalman smoother (EKS).

The constant velocity state-space model is defined by

$$\boldsymbol{f}_{\mathrm{CV}}(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{R}(\psi)\boldsymbol{\nu} + \boldsymbol{v} \tag{4.9a}$$

$$\boldsymbol{h}_{\mathrm{CV}}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{\eta} \\ r \end{bmatrix}, \tag{4.9b}$$

where $\boldsymbol{v}$ is the process noise. To get a satisfying result, the noise matrices where chosen as

$$Q = 10^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.10a}$$

$$R = 10^{-2} \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.10b}$$

4.3 Model Validation

By intuition, the instinctive approach of evaluating the estimated parameters would be to compare them to the true system parameters $\boldsymbol{\vartheta}_0$. However, when dealing with system identification the true parameters of the system are often unknown. Therefore, a method called cross validation (CV) is used. In CV, the model is validated with the validation data set $\mathcal{D}_v(N)$. The fit of the model to the validation data is calculated using the nomalized root mean square error (NRMSE)

$$\mathrm{NRMSE} = \sqrt{\frac{\sum_{k=1}^{N} \left(\boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k)\right)^2}{\sum_{k=1}^{N} \left(\boldsymbol{y}(k) - \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}(i)\right)^2}} \tag{4.11}$$

the model fit is then calculated as

$$\mathrm{fit}\left(\hat{\boldsymbol{y}}(k|\boldsymbol{\vartheta})\right) = 100\left(1 - \mathrm{NRMSE}\right). \tag{4.12}$$

This value can be interpreted as a percentage. The best possible fit will result in a value of 100. However, values below 0 are possible and there is no minimum value since the model can have arbitrarily bad fit (Ljungberg, 2020).

4.4 Experimental Platform

The experimental that was used platform was provided by Linköping University and can be seen in Figure 4.1. The propulsion system of the ship consists of three thrusters, two azimuth thrusters mounted in the aft and a tunnel thruster located in the bow of the ship. All thrusters are powered with brushless electrical motors and the rotation of the azimuth thrusters are powered by electrical servos. The ship was equipped with an IMU sensor that measures the angular velocities, linear accelerations as well as the magnetic field. To measure the position of the

(a) *(b)*

Figure 4.1: *The experimental platform used in the thesis. Seen from outside in (a) and the inside in (b).*

ship a GPS with RTK positioning is used and to measure the rotational speed of the thrusters RPM sensors are fitted to each thruster. The specifications of the model ship can be seen in Table 4.1.

Table 4.1: *Specifications of the model ship.*

Component	**Specifications**
Hull length	0.99 m
Hull width	0.30 m
Positioning	GPS with RTK positioning
Sensors	IMU and RPM sensors
Azipod, starboard	$l_{x,1} = -0.39\text{m}$, $l_{y,1} = 0.07\text{m}$
Azipod, portside	$l_{x,2} = -0.39\text{m}$, $l_{y,2} = -0.07\text{m}$
Bow thruster	$l_{x,t} = 0.37\text{m}$
Effective rudder area	$A_r = 0.001\text{m}^2$
Frontal projected area	$A_{\text{fw}} = 0.01\text{m}^2$
Lateral projected area	$A_{\text{lw}} = 0.1\text{m}^2$

4.5 Estimation Results

In this section, the results of the model estimation are presented. In Section 4.5.1, the results of the estimated model are given using simulated data. Thereafter, the result using experimental data are presented in Section 4.5.2.

4.5.1 Simulated Data

The grey-box model proposed in Chapter 2 is implemented in MATLAB with the parameters $\boldsymbol{\vartheta_0}$ given in Table 4.2. The parameters are chosen according to theory, to roughly match the model ship and the specifications of the model ship are presented in Section 4.4. The parameters μ_{az} and μ_t are assumed to be known from a Bollard pull test and are not estimated.

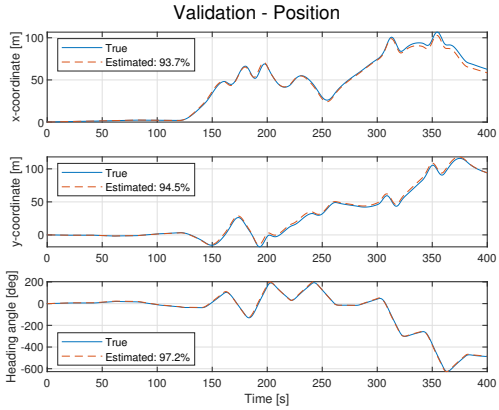
Then, the model was simulated for 400 seconds in SIMULINK with a 4th order Runge-Kutta method. A sampling time of 1 second was used, resulting in 400 data points. The input signals were chosen to generate an experiment similar to the one described in Appendix A.1. Also, white noise was added to the measurements. For the measurement of the position a standard deviation of $\sigma_{x,y} = \sqrt{0.01}$ m = 10 cm was used, meaning that it is a 95% probability that each measurement lies within a circle with radius 20 cm of the true value. The standard deviation for the measurement of the heading angle was chosen to be $\sigma_\psi = \sqrt{0.001}$ rad $\approx 1.81°$, implicating that the measured value lies within $\pm 3.62°$ of the true value with 95% probability. This was done twice, with slightly different input signals, to receive one dataset for estimation and another one for validation.

Thereafter, an EKS with a constant velocity model was applied to the estimation data to estimate the surge and sway velocities. The estimated states from the EKS were used as initial guesses for the decision variables and the initial guess for the parameters is presented in Table 4.2. With the initial decision vector, the NLP was formed and solved using the CasADi framework (Andersson et al., 2019) with the interior point optimizer (IPOPT) solver (Wächter and Biegler, 2005) to receive an estimate of the unknown parameters. The estimated parameters $\hat{\boldsymbol{\vartheta}}$ can be seen in Table 4.2. To evaluate the model, a cross validation with the validation data was performed with NRMSE as performance measure and the result can be seen in Figure 4.2.

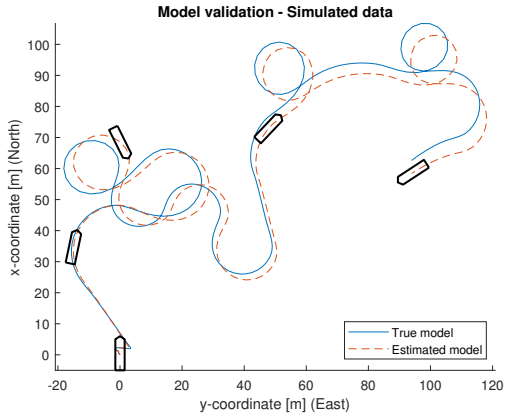
The fit of the model was over 93% for all DOF when using simulated validation data. Also, the estimated parameters lie close to the true values, indicating that the used input signal is sufficiently informative for the identification of the parameters. However, as can be seen in Table 4.2 there are some parameter estimates that are further from the true value than the initial guess. One reason for this can be that the poor initial guess of the non-linear damping parameters causes the optimization problem to adjust some parameters in the wrong direction. Another reason may be that the sampling time of 1 Hz does not give sufficiently informative data to capture all the dynamics of the vessel.

Table 4.2: *Parameters for the simulated grey-box model.*

Parameter	True value	Initial value	Estimated value	Unit
m_{11}	17.06	16.80	15.41	kg
m_{22}	17.41	17.50	15.50	kg
m_{33}	36.21	35.00	21.59	kgm^2
X_u	0.20	0.14	0.15	kg/s
$X_{\|u\|u}$	-0.79	0	-0.77	kg/m
Y_v	3.57	2.80	3.83	kg/s
Y_r	-51.19	-56.00	-49.08	kgm/s
$Y_{\|u\|v}$	0	-	-	kg/m
$Y_{\|v\|v}$	-0.71	0	0.71	kg/m
$Y_{\|r\|v}$	0	-	-	kg
N_v	4.29	7.00	4.19	kgm/s
N_r	21.59	21.00	18.07	kgm^2/s
$N_{\|v\|v}$	0	-	-	kg
$N_{\|r\|r}$	-6.24	0	-12.65	kgm^2
κ	$3.57 \cdot 10^{-8}$	0	$5.29 \cdot 10^{-8}$	s/m
C_N	0.20	0	0.20	-
V_w	1.53	0	1.16	m/s
β_w	78.69	0	81.23	°
c_x	0.70	0.55	0.50	-
c_y	0.80	0.75	0.95	-
c_n	0.10	0.15	0.05	-
μ_{az}	$1.30 \cdot 10^{-6}$	-	-	-
μ_t	$4.80 \cdot 10^{-8}$	-	-	-



(a)



(b)

Figure 4.2: *Validation of the model from simulated data. In (a) the validation of each DOF is presented separately, together they create the path in (b).*

4.5.2 Experimental Data

The experimental data was collected using the small-scale surface ship described in Section 4.4.

Firstly, a Bollard pull test were conducted to determine the thrust generated by each thruster. The test was performed in a small pool and the estimated parameters, μ_{az} and μ_t, can be seen in Table 4.3. More details about the test are available in Appendix A.2.

Unfortunately, no sensor to measure the angle of the azimuth thrusters was mounted to the ship. Therefore, a mapping from PWM to angle was carried out which is explained in Appendix A.3. Since this is not a measurement, it will be referred to as the calculated thruster angle.

The model identification experiment took place at Blå Lagunen, a small lake 10 kilometers south of Linköping. Before the experiments, a calibration of the magnetometer in the IMU was necessary to remove hard and soft iron distortions caused by magnetic material in the vicinity of the sensor. This was done by rotating the IMU in the Earth-frame, and since only the yaw angle is estimated, rotation about the z-axis was sufficient for this calibration. After the calibration, the control signals were chosen to generate a test similar to the one described in Appendix A.1. Data was collected at 1 Hz for the positioning, 5 Hz for the IMU and 10 Hz for the RPM sensors. During the tests, the weather conditions were mild, with wind speeds close to zero.

Before the estimation of the parameters, some preprocessing of the data was necessary. Firstly, the heading angle was estimated from the IMU data using a complementary filter (Mahony et al., 2008). Thereafter, all data was downsampled to 1 Hz, and outliers were removed. The preprocessed data can be seen in Appendix A.

Similarly as for the simulated data, an EKF with a constant velocity model along with a RTS smoother was applied to the position data and the estimated heading angle to estimate the surge and sway velocities of the ship to use as an initial guess in the parameter estimation.

During the parameter estimation a problem arised of getting the NLP to converge. This was solved by simplifying the model, at first the rudder model was removed due to its complexity and limited effect on the ship. Then, the wind model was disregarded, mostly since the wind speed during the tests was close to zero. Also, the parameter κ was set to zero to help convergence. Furthermore, it was found out that the PWM to angle mapping did not suffice, causing the ship to turn even when the calculated angle was zero, as seen in Figure 4.3. Therefore, the azimuth angles were chosen to be a decision variable at each control interval in the NLP, and was thus also included in the objective function

$$\Phi(\boldsymbol{w}, \bar{\boldsymbol{y}}) = \sum_{i=1}^{N} \left(\bar{\boldsymbol{y}}_i - \boldsymbol{y}_i\right)^T \boldsymbol{W} \left(\bar{\boldsymbol{y}}_i - \boldsymbol{y}_i\right) + \sum_{k=1}^{K} \sum_{i=1}^{2} \left(\boldsymbol{\alpha}_{k,i} - \bar{\boldsymbol{\alpha}}_{k,i}\right)^T \boldsymbol{W}_\alpha \left(\boldsymbol{\alpha}_{k,i} - \bar{\boldsymbol{\alpha}}_{k,i}\right) ,$$

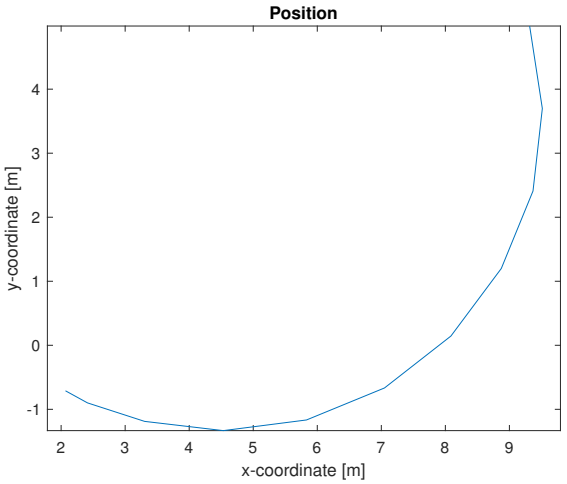


Figure 4.3: *The resulting path when the thruster angles were zero according to the PWM to angle mapping. As seen, the ship turns with a radius of about 5 meters.*

where K is the number of control intervals.

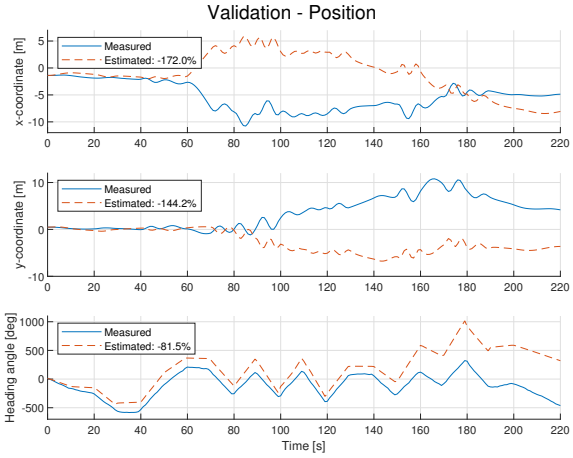
Thereafter, the NLP was formed and solved in two steps using the CasADi framework (Andersson et al., 2019) with the IPOPT solver (Wächter and Biegler, 2005). Firstly, without the Coriolis term $\boldsymbol{C}(\boldsymbol{\nu})$ to get a good initial guess of the grey-box parameters. Then, different combinations of the damping parameters were tested to obtain a combination that converged. The result from the first step was then used as an initial guess when the Coriolis term was included in the second step. Lastly, as many of the damping parameters were removed, while still getting the optimization problem to converge. The estimated parameters $\hat{\boldsymbol{\vartheta}}$ as well as the initial guesses can be seen in Table 4.3.

To validate the model, validation data was collected in the same manner, with slightly different control signals. The model was thereafter simulated using a RK4 method with the measured control signal. The results are presented in Figure 4.4. As can be seen in Figure 4.4, the fit of the model to the measurements is really poor. However, it can be seen that the ship moves in a similar way, indicating that the model is better than the fit measure implies. This can be seen by looking at the generalized velocities of the ship in Figure 4.5, where the velocities estimated by the EKS match the velocities of the simulated data using the estimated model quite well. Also, a small deviation in the heading angle causes a big drift in the position in the long run.

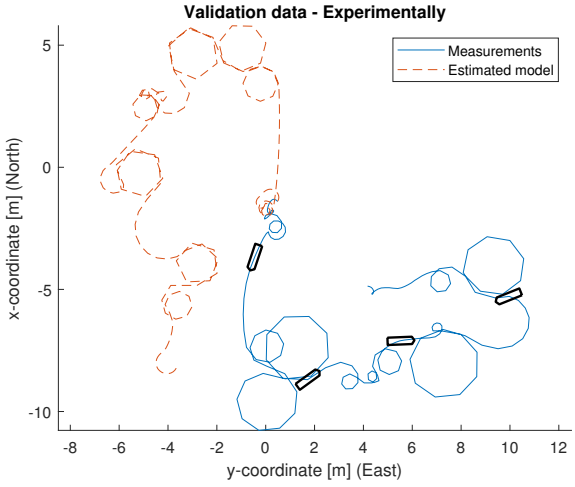
To compensate for the deviation of the azimuth angles, an NLP was formed where the estimated parameters were fixed and the azimuth angles chosen as decision variables. After the NLP was solved, the model was once again simulated with a

Table 4.3: *Parameters for the grey-box model of the model ship. In the second column, the initial values for the first NLP are presented. The result of the first NLP is thereafter used as initial values for the second NLP, as presented in the second column. In the forth column, the estimated parameter values of the second NLP are presented, as well as the estimated parameters from the Bollard pull test.*

Parameter	Initial value without $\boldsymbol{C(\nu)}$	Initial value with $\boldsymbol{C(\nu)}$	Estimated value	Unit
m_{11}	16.8	7.39	14.10	kg
m_{22}	25.2	8.33	14.10	kg
m_{33}	42.0	0.55	0.23	kgm^2
X_u	1.4	14.40	0.98	kg/s
$X_{\lvert u\rvert u}$	0	0	-10.28	kg/m
Y_v	1.4	$4.20 \cdot 10^{-8}$	$4.52 \cdot 10^{-9}$	kg/s
Y_r	-42	0.76	-1.3	kgm/s
$Y_{\lvert u\rvert v}$	0	-7.82	-47.90	kg/m
$Y_{\lvert v\rvert v}$	0	-24.92	-17.14	kg/m
$Y_{\lvert r\rvert v}$	0	-13.66	-1.32	kg
N_v	1.4	-4.24	2.21	kgm/s
N_r	21	2.65	2.09	kgm^2/s
$N_{\lvert v\rvert v}$	0	-20.77	9.09	kg
$N_{\lvert r\rvert r}$	0	$-5.64 \cdot 10^{-9}$	$-2.31 \cdot 10^{-10}$	kgm^2
μ_{az}	-	-	$1.435 \cdot 10^{-6}$	-
μ_t	-	-	$4.840 \cdot 10^{-8}$	-



(a)



(b)

Figure 4.4: *Validation of the model from experimental data without correction of the thruster angles. In (a) the validation of each DOF is presented separately, together these create the path in (b).*

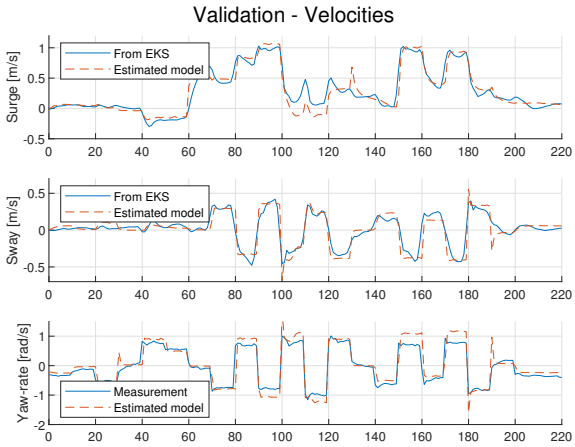
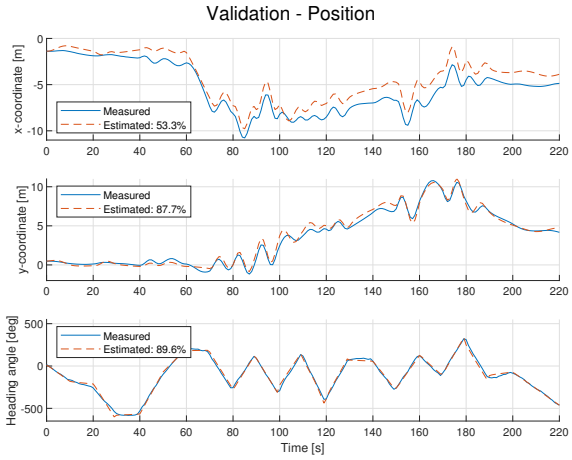


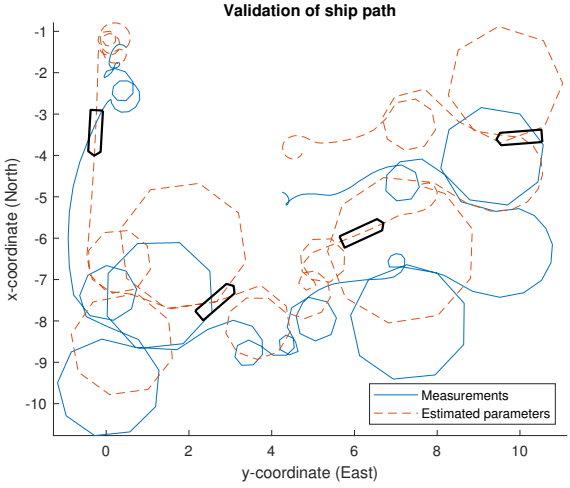
Figure 4.5: *Validation of the generalized velocities of the ship using the experimental data without correction of the thruster angles.*

RK4 method using the estimated control signal. The results can be observed in Figure 4.6, where the corrected azimuth angles are presented in Figure 4.7.

By allowing the NLP to slightly change the azimuth angles, a much better result is achieved as shown in Figure 4.6. With a model fit close to 90 percent for y and ψ. However, the x value is a bit off at roughly 50 percent. It seems like this is caused by the first 50 seconds of the path, and the ship is constantly off by around one meter afterwards. Likely, this is a hard manoeuver to model since the boat mostly just spun around. Another possible effect on the fit is the wind during the experiment. Although the conditions were good, some wind guts could have put the ship in the wrong way.



(a)



(b)

Figure 4.6: *Validation of the model from experimental data with correction of the thruster angles. In (a) the validation of each DOF is presented separately, together these create the path in (b).*

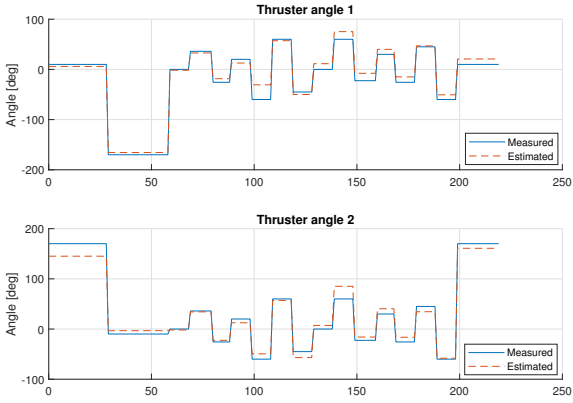


Figure 4.7: *The calculated angles of the azimuth thrusters of the validation data, as well as the corrected angles from the solution of the NLP.*

5
Path Planning

The goal with a path planner is to find a feasible path between a start and goal state. Let $\mathcal{X}$ be the state space of a given area, not to be confused with the state-space model. With $\mathcal{X}_{\text{obst}}$ defined as the set containing the static obstacles, the free-space in which a path is sought is given by $\mathcal{X}_{\text{free}} = \mathcal{X} \backslash \mathcal{X}_{\text{obst}}$ assuming no dynamic obstacles.

In this chapter, a graph search approach using an undirected graph (roadmap) created from a Voronoi diagram is explored. A discretization $\mathcal{X}_r$ of $\mathcal{X}_{\text{free}}$ is done in the form of a roadmap over a static environment $\mathcal{X}$ and an auxiliary path is found using a graph search algorithm. The auxiliary path is then smoothened to decrease the impact of discretization artifacts.

In Section 5.1, the theory and method for creating an undirected graph (roadmap) using a Voronoi diagram are given. Methods of how vertices such as start and goal position can be inserted to the roadmap are presented in Section 5.2. Section 5.3 contains the methods used for graph search and path smoothing to obtain an auxiliary path that will be used to initialize the trajectory planning in Chapter 6.

5.1 Roadmap

A roadmap can be created given that the environment is assumed to be static (LaValle, 2006). There exist multiple methods to obtain roadmaps, one of which is the generalized Voronoi diagram (GVD), also known as maximum-clearance roadmap (LaValle, 2006). As the name suggest, this type of roadmap aims at maximizing the distance to obstacles, thus reducing the risk of collision.

5.1.1 Voronoi Diagram

The ordinary Voronoi diagram is constructed from a set of distinct points $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$ in the plane, called generator points (Sugihara, 1993). For each generator point $\boldsymbol{p}_i$ there is a Voronoi region

$$V(\boldsymbol{p}_i) = \{\boldsymbol{p} \mid d(\boldsymbol{p}, \boldsymbol{p}_i) < d(\boldsymbol{p}, \boldsymbol{p}_j) \quad \text{for any} \quad j \neq i\} \tag{5.1}$$

such that for any point $\boldsymbol{p} \in V(\boldsymbol{p}_i)$ the closest generator point is $\boldsymbol{p}_i$, where $d(\boldsymbol{p}, \boldsymbol{q})$ is the Euclidean distance between two points $\boldsymbol{p}$ and $\boldsymbol{q}$ (Sugihara, 1993). The Voronoi diagram generated from P is then denoted $\mathscr{V}(P)$. The lines splitting the Voronoi regions are called Voronoi edges and a point connecting Voronoi edges is called a Voronoi vertex (Sugihara, 1993).

5.1.2 Generalized Voronoi Diagram (GVD)

The generalized Voronoi diagram for figures (obstacles) is similar to the ordinary Voronoi diagram, but instead of a set of generator points P there is a set of nonoverlapping generator obstacles $G = \{g_1, g_2, \ldots, g_n\}$, where the obstacle $g_i \subset \mathbb{R}^2$ is a closed set of points for $i = 1, 2, \ldots, n$, and $g_i \cap g_j = \emptyset$ for $i \neq j$ (Sugihara, 1993). Similar to (5.1), the Voronoi region for each obstacle g_i is

$$V(g_i) = \{\boldsymbol{p} \mid d(\boldsymbol{p}, g_i) < d(\boldsymbol{p}, g_j) \quad \text{for any} \quad j \neq i\} \tag{5.2}$$

An approximation of the GVD is proposed by Sugihara (1993). The approximation is given by first generating the ordinary Voronoi diagram for a set of points approximating the boundary of the obstacles and then removing edges between Voronoi regions where both generator points belong to the same obstacle.

For a Voronoi vertex to be generated between two obstacles the distance y in Figure 5.1 has to be larger than zero. With

$$y = \frac{x^2 - ab}{2x}, \tag{5.3}$$

then $x > \sqrt{ab}$ and the worst case is when $a = b = d/2$, i.e., D is equidistant from A and B, for proofs see Appendix B.1 and Appendix B.2.

By using (5.3) in the worst case scenario ($a = b = d/2$), the distance y in Figure 5.1 is

$$y = \frac{x^2 - (d/2)^2}{2x}, \tag{5.4}$$

and thus $y \to x/2$ when $d \to 0$ and $y < 0$ when $d > 2x$, i.e., the Voronoi vertex is either inside of g_i or on the other side of g_i. From (5.2), it can be said that $\boldsymbol{p} \in V(g_i)$ is closer to obstacle g_i than any other obstacle g_j, $j \neq i$. Thus, the Voronoi vertices are equidistant to two or more obstacles in an ideal GVD and $y = x/2$. However, since the distance between generator points on each obstacle in G then would have to be $d = 0$, this is impossible. Instead d can be calculated to guarantee that a Voronoi vertex is not generated closer to an obstacle than the

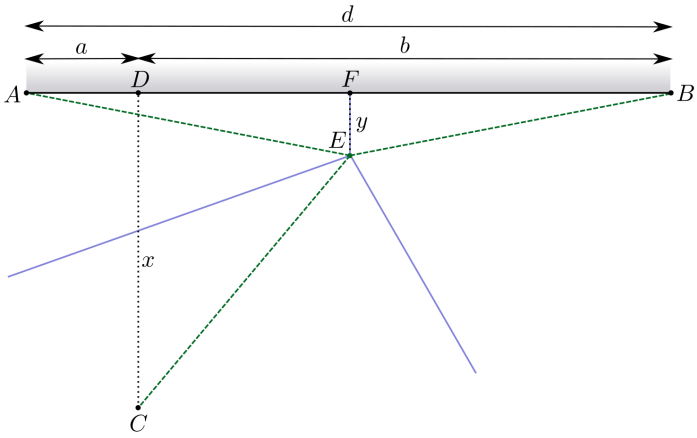


Figure 5.1: *Voronoi diagram (blue) generated by the points A, B and C, where generator points A and B belongs to the same obstacle and generator point C belongs to another obstacle. The green dotted lines are of equal length, which indicates that the Voronoi vertex E is equidistant from the generator points A, B and C.*

distance d_c. Rewriting (5.4) with the requirement $y \geq d_c$ gives

$$d \leq 2\sqrt{x^2 - 2xd_c}, \quad \text{where} \quad d_c \in [0, x/2) \quad \text{and} \quad d > 0. \tag{5.5}$$

Equation (5.5) in turn can be rewritten to return a lower limit on the distance x between two obstacles that will guarantee that a Voronoi vertex is no closer to an obstacle than d_c

$$x \geq \frac{1}{2}\sqrt{d^2 + 4d_c^2} + d_c. \tag{5.6}$$

5.1.3 Generate Roadmap from GVD

An algorithm for obtaining an approximate GVD from figures (obstacles) is proposed by Sugihara (1993). This algorithm however, does not leave any edges in $g_i^b = g_i^c \backslash g_i$, where g_i^c is the convex hull of obstacle g_i and thus, g_i^b is the free-space between the convex hull and the obstacle. An illustration of the remaining edges is given in Figure 5.2(a). These edges in g_i^b are important to keep since the Voronoi edges and vertices later form the roadmap. To solve this problem, only edges that are intersecting with an obstacle (or that are completely inside an obstacle) are removed, see Figure 5.2(b).

As can be seen in Figure 5.2, some edges go to infinity and there are no edges that surround the individual obstacles. To solve this either generator points can be placed on a bounding box encompassing the obstacles or along the edge of each expanded obstacle. The first method however, only works for the outer obstacles

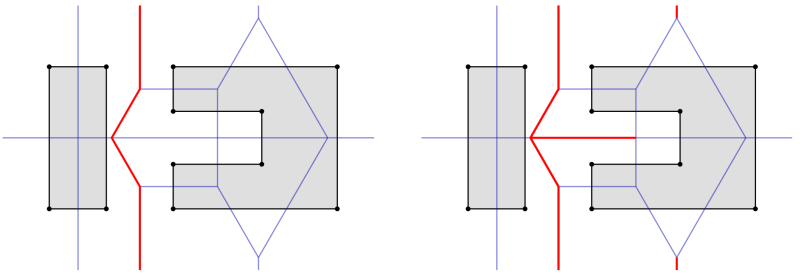


Figure 5.2: *Voronoi diagram (blue) generated from the corners of the obstacles (black dots). The red edges are what is returned by the algorithm given by (Sugihara, 1993) (a) and the modified algorithm (b) where edges that are intersecting obstacles are removed.*

(the ones closest to the bounding box) while the obstacles in the middle of the area still can have edges really far away. Thus the second method is used.

To get edges enclosing each individual obstacle, an expanded obstacle is created and generator points are distributed along its boundary, see Figure 5.3. The distance $d_{i,e}$ with which obstacle g_i is expanded by is calculated using (5.6), where x is replaced with $d_{i,e}$

$$d_{i,e} \geq \frac{1}{2}\sqrt{d_{i,\max}^2 + 4d_c^2} + d_c. \tag{5.7}$$

Here, $d_{i,\max}$ is the largest distance between two generator points on g_i and d_c is the closest possible distance between an obstacle and a Voronoi vertex generated by the obstacle and its expansion.

However, as visible in Figure 5.3(a), when the expanded obstacles overlap, the guarantee from (5.7) does not hold. Thus, generator points on the expanded obstacles that lie within another expanded obstacle must be removed, see Figure 5.3(b). Then, whether or not a Voronoi vertex is generated closer than d_c to an obstacle only depends on the resolution (the distance d between generator points) of the obstacles and the distance between them. Even more generator points on the expanded obstacles can be removed to reduce patterns on the modified GVD (MGVD) like the one between the obstacles in Figure 5.3(c). If this pattern is undesired, generator points on the expanded obstacles that overlap any of the other obstacles expanded even further than $d_{i,e}$ can be removed.

A comparison of a roadmap generated without generator points on expanded obstacles and with generator points on expanded obstacles is shown in Figure 5.4. As can be seen in Figure 5.4(b), there are roads (edges) closer to the obstacles and thus the best path between two points is more likely to be shorter, since the path could be closer to a straight line between the points. But the main advantage with the generator points on the expanded obstacles is the addition of road

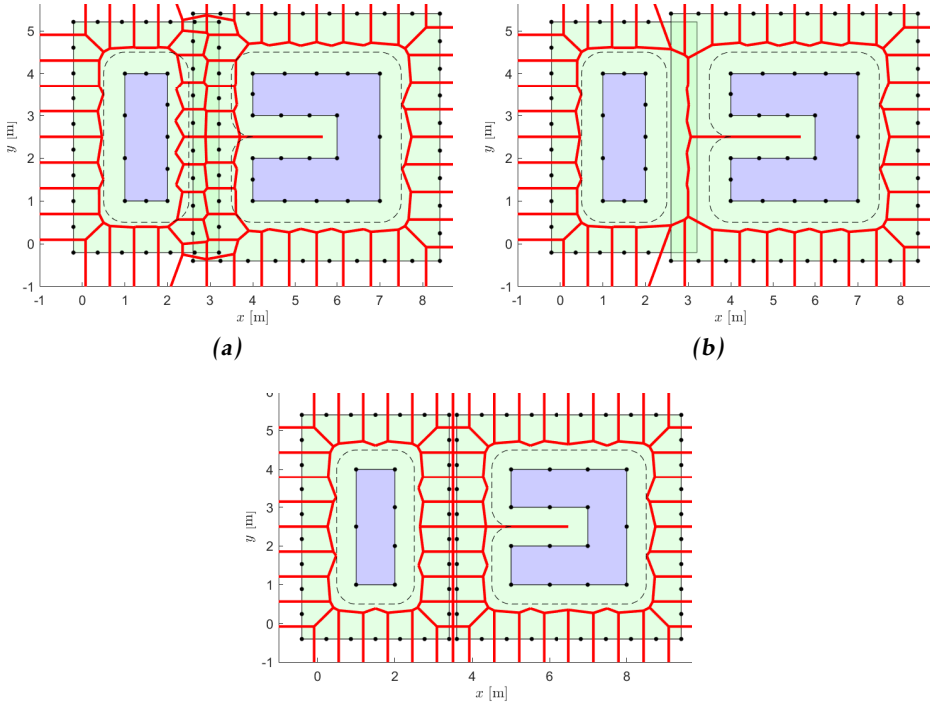
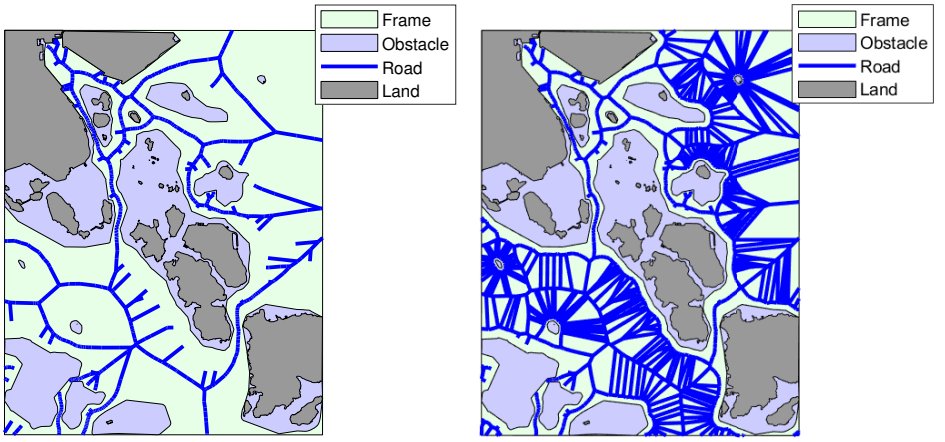


Figure 5.3: *Modified GVD (red) with no generator points (black dots) removed (a) and with generator points on expanded obstacles (green) intersecting another expanded obstacle removed (b). In (c) a different map is used compared to the one in (a) - (b), to show what happens if the extended obstacles does not intersect. The dashed lines marks the original obstacles expanded by d_c in* (5.7).



(a) *(b)*

Figure 5.4: *Example of roadmap without generator points on expanded obstacles (a) and with generator points on expanded obstacles (b).*

connections over areas where there were none before.

The resolution of the obstacles d can be determined for individual line segments of each obstacle. Depending on the shortest distance x from a line segment on an obstacle to all other obstacles, d may be decreased, i.e., the resolution may be increased. Given that a certain minimum distance d_s between Voronoi vertices and obstacles is desired, e.g. a safe distance that a vehicle needs to keep to obstacles, the resolution of generator points can be calculated adaptively with respect to d_s. If $x < 2d_s$ there is no meaning in adding generator points on that line segment since the vehicle would not be allowed to travel there. With $x \geq 2d_s$ however, then d may be decreased on that line segment to make the MGVD smoother and the edges of the GVD end up closer to the middle of the free-space between obstacles in that area.

A bounding box (minimum area rectangle) is fitted to the obstacles which can be said to be the boundary of $\mathcal{X}$. Edges completely outside the bounding box are removed while those crossing its boundary are shortened, i.e., the outer point is moved to the intersection between the bounding box and the edge. This is done to remove outliers from the MGVD that would not add anything to the roadmap.

Inspired by the algorithm for approximate GVD from figures given in Sugihara (1993), the final algorithm for obtaining the roadmap is given in Algorithm 5.1 and the result can be observed in Figure 5.5.

Let $\mathcal{V}$ be the vertices of $\mathcal{R}$ from Algorithm 5.1. The discrete state space is then

Algorithm 5.1: Roadmap from obstacles

Input: Set $G = \{ g_1, g_2, \ldots, g_n \}$ of n nonoverlapping obstacles.
Output: Roadmap $\mathscr{R}$ from obstacles

1 Create the minimum area rectangle B for the obstacles
2 For each $i = 1, 2, \ldots, n$, create a finite set P_i of points that approximates the boundary of g_i. Also create an expanded obstacle g_i^e of g_i and create a finite set of points P_i^e that approximates its boundary. Remove from P_i^e those points that lie within any of the expanded obstacles g_j^e, $j \neq i$.
3 Construct the ordinary Voronoi diagram $\mathscr{V}$ generated by point set $P_1 \cup P_2 \cup \cdots \cup P_n \cup P_1^e \cup P_2^e \cup \cdots \cup P_n^e$
4 Delete from $\mathscr{V}$ those Voronoi edges crossing the boundary of an obstacle or that are completely inside an obstacle.
5 Remove the edges that are outside B and shorten edges that intersect its boundary so that the edge ends at the intersection between the boundary of B and the edge. Finally, delete isolated Voronoi vertices, if any exist, from $\mathscr{V}$.
6 Return roadmap (graph) $\mathscr{R} = \mathscr{V}$.

given by

$$\mathcal{X}_r = \{ \boldsymbol{p} \mid \boldsymbol{p} \in \mathcal{V} \}, \quad \text{where} \quad \boldsymbol{p} = \begin{bmatrix} x & y \end{bmatrix}^T \tag{5.8}$$

5.2 Connecting Start and Goal Positions

With a roadmap available, start and goal positions can be connected to the map. In this section, three possible methods of connecting vertices to the roadmap are explored. The first method connects an added vertex to the closest point on the roadmap. The second method is similar to the first method but connects to the closest point on each edge of the roadmap given that the connecting edge only intersects the roadmap once, i.e., the connection cannot cross any edge already in the roadmap. The third method makes connections from an added vertex to feasible crossroads on the roadmap. A crossroad refers to a vertex in the roadmap that is connected to three or more other vertices. For a visualization of how the different methods connect to the roadmap, see Figure 5.6.

5.2.1 Connecting to Closest Point

In the first method, a vertex $\boldsymbol{v}$, e.g. start or goal position, that is added to the map is connected to the point $\boldsymbol{p}_c$ on the map that is closest to $\boldsymbol{v}$ such that the straight line between $\boldsymbol{v}$ and $\boldsymbol{p}_c$ is collision-free and no closer than d_s to any obstacle. This is done by projecting $\boldsymbol{v}$ on all edges in the roadmap and then trying to connect $\boldsymbol{v}$ to the projection with shortest distance to $\boldsymbol{v}$. If the connection is collision-free and the minimum distance to obstacles is satisfied $\boldsymbol{p}_c$ is found. Thus $\boldsymbol{v}$ may not be

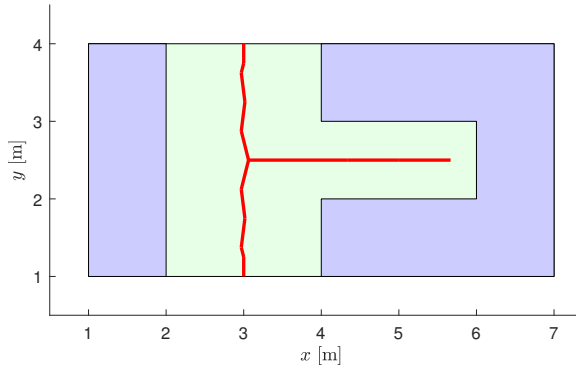
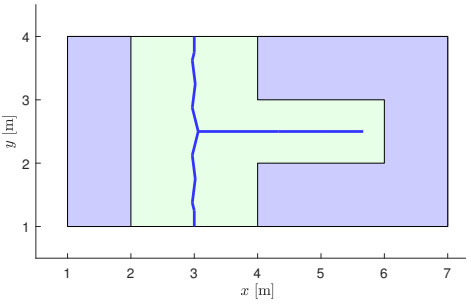
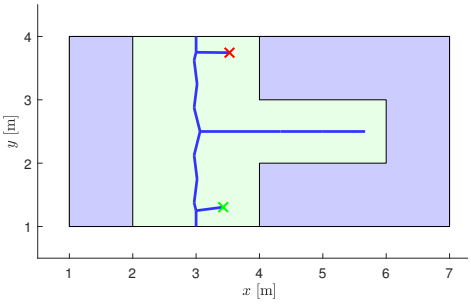


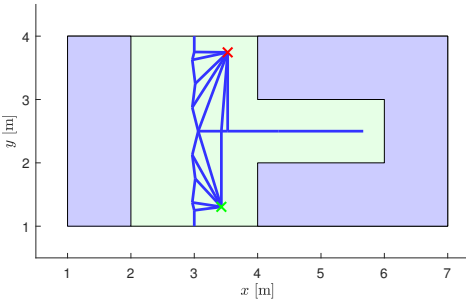
Figure 5.5: *Roadmap (red) using Algorithm 5.1, with obstacles (blue) and minimum area rectangle (green ∪ blue).*



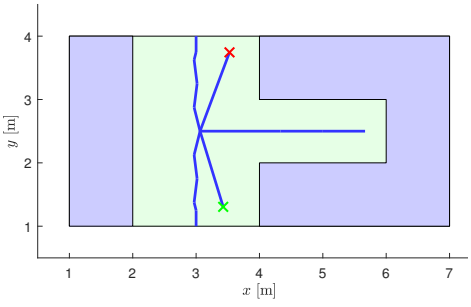
(a) *Original roadmap without added vertices*



(b) *Connecting to Closest Point*



(c) *Connecting to Nearby Edges*



(d) *Connecting to Crossroads*

Figure 5.6: *Example of how the methods connect to the roadmap.*

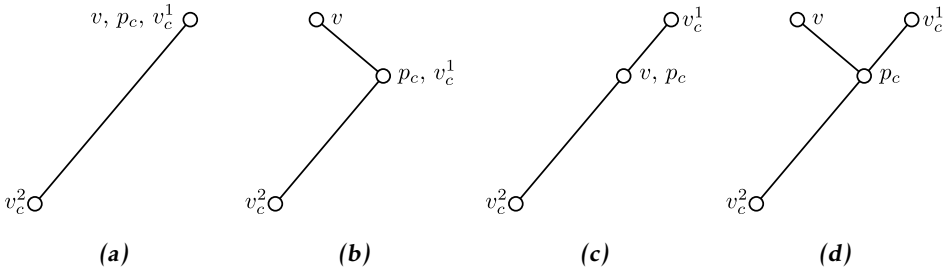


Figure 5.7: *Different ways for $\boldsymbol{v}$ to be connected to the roadmap, where $\boldsymbol{v}_c^1$ and $\boldsymbol{v}_c^2$ are vertices in the map. (a) Case I: $\boldsymbol{v} \in \mathcal{X}_r$, (b) Case II: $\boldsymbol{v} \notin \mathcal{X}_r$ but $\boldsymbol{p}_c \in \mathcal{X}_r$, (c) Case III: $\boldsymbol{v} = \boldsymbol{p}_c$ and $\boldsymbol{v}$, $\boldsymbol{p}_c \notin \mathcal{X}_r$, (d) Case IV: $\boldsymbol{v} \neq \boldsymbol{p}_c$ and $\boldsymbol{v}$, $\boldsymbol{p}_c \notin \mathcal{X}_r$.*

connected to the closest point on the map, but rather the closest possible point in regards to the obstacles. There are four possible cases when a vertex is connected to the roadmap and they are shown in Figure 5.7.

Case I This occur when $\boldsymbol{v} \in \mathcal{X}_r$, i.e., $\boldsymbol{v}$ already exist in the roadmap. Then $\boldsymbol{v}$ does not need to be inserted.

Case II When $\boldsymbol{v} \notin \mathcal{X}_r$ and $\boldsymbol{p}_c \in \mathcal{X}_r$. Then insert $\boldsymbol{v}$ and add an edge between $\boldsymbol{v}$ and $\boldsymbol{p}_c$.

Case III If $\boldsymbol{v} = \boldsymbol{p}_c$ and $\boldsymbol{v}$, $\boldsymbol{p}_c \notin \mathcal{X}_r$, then $\boldsymbol{v}$ is on an edge in the roadmap. Let $\boldsymbol{v}_c^1$ and $\boldsymbol{v}_c^2$ denote the endpoints of said edge. Then insert $\boldsymbol{v}$ and split the edge into two, one between $\boldsymbol{v}_c^1$ and $\boldsymbol{v}$ and one between $\boldsymbol{v}_c^2$ and $\boldsymbol{v}$.

Case IV When $\boldsymbol{v} \neq \boldsymbol{p}_c$ and $\boldsymbol{v}$, $\boldsymbol{p}_c \notin \mathcal{X}_r$ both $\boldsymbol{v}$ and $\boldsymbol{p}_c$ are inserted. Since $\boldsymbol{p}_c$ lies on an edge with endpoints $\boldsymbol{v}_c^1$ and $\boldsymbol{v}_c^2$ in the roadmap, split that edge as in Case III. Also an edge between $\boldsymbol{v}$ and $\boldsymbol{p}_c$ has to be added.

5.2.2 Connecting to Nearby Edges

The second connection method, connecting to nearby edges, is similar to the previous one but instead of connecting only to one point on the roadmap, it connects to all edges where the line between the added vertex $\boldsymbol{v}$ and $\boldsymbol{p}_c$ does not intersect any other edges.

5.2.3 Connecting to Crossroads

Lastly, connecting to crossroads is a method were a vertex $\boldsymbol{v}$ is connected to crossroads on the roadmap. For $\boldsymbol{v}$ to be able to connect to a crossroad $\boldsymbol{v}_{cr}$ the line segment L formed by $\boldsymbol{v}$ and $\boldsymbol{v}_{cr}$ cannot intersect any obstacle and the distance between an obstacle and L has to be greater or equal to d_s.

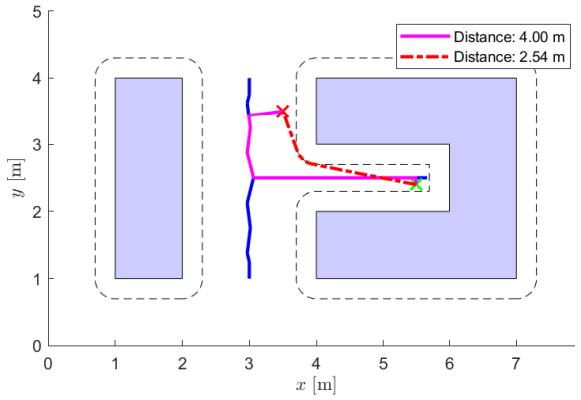


Figure 5.8: *The initial path from A* (magenta line) with sharp corner and smoothened path (dashed red line) without sharp corners, from start position (green cross) to goal position (red cross). Black dashed lines are marking the minimum distance allowed to the obstacles.*

5.3 Auxiliary Path

An auxiliary path is obtained through a couple of steps: graph search and path smoothing. A graph search algorithm is employed to a graph (roadmap) to quickly find a shortest path in the roadmap. The path is then smoothened to reduce discretization artifacts such as, sharp turns and zig-zag patterns, which will also lead to reduced path length.

5.3.1 Graph Search

A graph search algorithm can be applied to the roadmap described in Section 5.1 to find the shortest path between two points. An algorithm that is suitable for this is the A-star (A*) algorithm since it is both fast and guarantees optimality (Hart et al., 1968). The heuristic function used in A* is the Euclidean distance in two dimensions, which gives optimality in terms of distance. The acquired path using A* may, however, contain sharp corners due to the discretization of the free-space, see Figure 5.8, and since the GVD ideally maximizes the distance to obstacles the path may also be unnecessarily long (Patle et al., 2019). Thus, the path can often be shortened without increasing the risk significantly. In the A* algorithm, an extra condition, checking if a vertex is closer than a given safe distance to any obstacle, was added to prevent the algorithm from finding paths going too close to obstacles. This is needed since parts of the free-space in the map that is used to generate the roadmap can be narrow. Thus, the roadmap can contain road segments where the safe distance is not satisfied.

5.3.2 Path Smoothing

An iterative method for removing redundancies and smoothen the path while at the same time keeping a given minimum distance to obstacles is given by Bhattacharya and Gavrilova (2007). Since the smoothened path will not get closer to any obstacle than the given minimum distance (Bhattacharya and Gavrilova, 2007), it will remain within $\mathcal{X}_{\text{free}}$.

Let $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k\}$ be the vertices (in sequence) of a path. Redundancies (unnecessary vertices) in the path are removed by checking if vertex $\boldsymbol{v}_i$ can be connected to $\boldsymbol{v}_{i+2}$. If they can be connected, i.e., the line segment L between $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+2}$ is collision-free and the shortest distance from L to an obstacle is greater than d_s, then $\boldsymbol{v}_{i+1}$ is said to be redundant and is removed. When the whole path has been checked, if the number of vertices in the path with removed redundancies is the same as the number of vertices in the original path, all redundancies have been removed, otherwise do the same for the new path. This step is done to increase the performance of the path smoothing algorithm.

After the redundancies have been removed the path can still contain sharp corners. Thus, the next step is to smoothen the path. The smoothing algorithm given in Bhattacharya and Gavrilova (2007) uses a corner cutting technique and is interpreted as can be seen in Algorithm 5.2 with a slight modification. With this method sharp, corners are effectively removed and the initial path is shortened, see Figure 5.8.

A modification was introduced (Line 10 - Line 13) to address a problem that occurred when the incident edges of a vertex were almost parallel and is illustrated in Figure 5.9. When it is checked if the path can be smoothened around the vertex marked in Figure 5.9(a), an endpoint is reached in the first step. Since a connection could be made, the marked vertex is replaced with the point marked with a circle. Then, the vertex marked in Figure 5.9(b) is checked and replaced with the point marked as a circle since the endpoint was reached on one of the incident edges and a connection was possible. After that, the path would be almost the same as in Figure 5.9(a) and a infinity loop is created.

Algorithm 5.2: Path smoothing

Input: A path $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k\}$, a maximum resolution $\Delta_{\min}$ and a minimum clearance d_s
Output: A new path $\mathcal{V}$

$N \longleftarrow$ # vertices in $\mathcal{V}$
if $N = 2$ **then**
  return
end
$\Delta \longleftarrow$ mean of the distance between vertices in $\mathcal{V}$
while $\Delta > \Delta_{min}$ **do**
  $\Delta \longleftarrow \Delta/2$
  $i \longleftarrow 2$ `// Start with second vertex`
  while $i \leq N - 1$ **do**
    if $\overline{\boldsymbol{v}_i\boldsymbol{v}_{i-1}}$ *and* $\overline{\boldsymbol{v}_i\boldsymbol{v}_{i+1}}$ *parallel* **then**
      $i \longleftarrow i + 1$
      continue
    end
    $\boldsymbol{e}_1 \longleftarrow$ evenly distributed points with spacing Δ between $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i-1}$
    $\boldsymbol{e}_2 \longleftarrow$ evenly distributed points with spacing Δ between $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$
    `// The first point in` $\boldsymbol{e}_1$ `and` $\boldsymbol{e}_2$ `is` $\boldsymbol{v}_i$ `and the last points are` $\boldsymbol{v}_{i-1}$ `and` $\boldsymbol{v}_{i+1}$ `respectively`
    $j \longleftarrow 2$
    $C \longleftarrow \emptyset$
    $M \longleftarrow$ the least number of points among $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$
    while $j \leq M$ **do**
      $L \longleftarrow$ line segment between $\boldsymbol{e}_1[j]$ and $\boldsymbol{e}_2[j]$
      if L *is collision-free and clearance to* L *is greater than* d_s **then**
        $C \longleftarrow L$
        $j \longleftarrow j + 1$
      else
        break
      end
    end
    if C *is empty* **then**
      $i \longleftarrow i + 1$
    else
      $\mathcal{V} \longleftarrow$ replace $\boldsymbol{v}_i$ with $C \backslash \mathcal{V}$
      if *endpoint of* $\boldsymbol{e}_1$ *is reached and* $i \neq 2$ **then**
        $i \longleftarrow i - 1$
      end
    end
    $N \longleftarrow$ # vertices in $\mathcal{V}$
  end
end

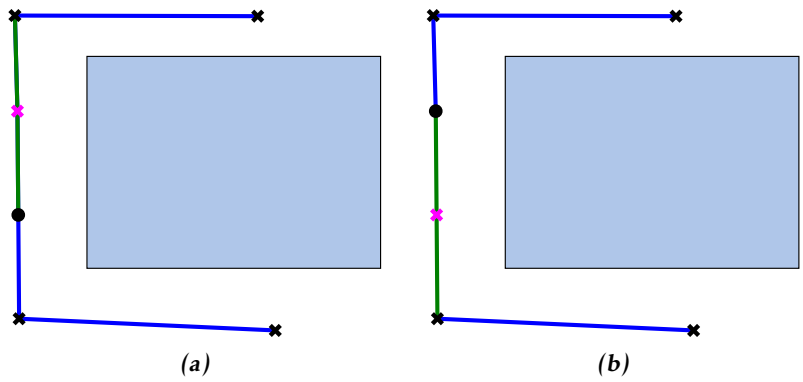


Figure 5.9: *Illustration of the problem discovered in the interpretation of the smoothing algorithm given by Bhattacharya and Gavrilova (2007). The blue rectangle is an obstacle, the blue line is a path acquired from removing redundancies. Vertices are marked as crosses and the black circle is a temporary point used by the smoothing algorithm. (a) - (b), Path is smoothened around the magenta vertex. The green line indicates the edge replacing the magenta vertex, i.e., the magenta vertex is replaced by the black circle.*

5.4 Path Planning Results

For the results the maps have been generated using $d_s = 15/16$. The resolution on each individual edge of an obstacle is chosen using (5.5) with $d_c = 0.95x/2$. For the map generated with generator points on expanded obstacles (WEO) the expanded obstacle g_i^e is expanded using (5.7). Generator points on g_i^e are distributed with a maximum distance $d_{i,e}/2$. Generator points on an expanded obstacle g_j^e are removed if they are within any of the other obstacles g_i expanded with distance $2d_{i,e}$ for $i \neq j$. The safe distance for graph search, redundancy removal and path smoothing was set to 0.3.

5.4.1 Roadmap Creation

A roadmap with generator points on expanded obstacles (WEO) and without (NEO), was created ten times each for both types (WEO and NEO) with the same obstacles. The average computational times are presented in Table 5.1 and as seen the WEO map takes approximately twice as long to be calculated which is a consequence of adding more generator points and thus more Voronoi edges to check in the roadmap creation. The created NEO map can be seen in Figure 5.4(a) while the WEO map is illustrated in Figure 5.4(b).

Table 5.1: Mean computational time [s] for roadmap creation.

WEO	NEO
16.06	8.76

Table 5.2: Time [ms] of each connection method for the different missions and map types.

Map Type		WEO			NEO		
Method		CC	CNE	CCR	CC	CNE	CCR
Mission	1	54.5	11920.0	4564.0	18.7	4157.3	393.9
	2	28.4	9280.5	3260.7	17.1	3838.9	378.0
	3	29.9	6277.3	2119.5	19.8	2953.8	296.3
	4	22.4	9468.6	3846.3	17.3	3208.2	326.3
	5	34.1	6703.7	2518.4	17.6	2558.9	297.2
	6	24.2	5608.0	1814.6	16.4	2834.1	286.8
	7	23.3	6073.2	2230.5	17.4	2546.0	243.8
	8	27.9	8905.5	3023.0	15.4	3901.5	378.9
	9	20.9	8387.1	3440.0	16.9	2624.7	282.2
	10	20.7	7898.1	3206.8	16.2	2710.9	273.4
Mean		28.6	8052.2	3002.4	17.3	3133.4	315.7

5.4.2 Connecting Methods

The connection methods examined are described in Section 5.2 and are: Connecting to Closest Point (CC), Connecting to Nearby Edges (CNE) and Connecting to Crossroads (CCR). The data was collected from ten different missions on both types of roadmaps, created with generator points on expanded obstacles (WEO) and without (NEO). A mission is defined as going from a starting position to a goal position.

In Table 5.2, the time for connecting both start and goal position to the roadmap is presented for the missions. Table 5.3 contains the length of the auxiliary path, i.e., the smoothened path, given the connection method.

As can be seen in Table 5.2, the connecting method demanding the least amount of computational power is the CC method followed by CCR and CNE. The computation time increases for the WEO map type compared to NEO due to the increased number of edges in the map, hence increasing the number of calculations and collision checks needed to ensure a feasible connection. When connecting a vertex using the CNE or CCR method, a significant number of collision checks are performed, causing them to be slower than the CC method. The CCR method is faster than CNE since it only has to check for collisions with obstacles while CNE has to check both for collisions with obstacles and intersections with other edges in the roadmap.

Table 5.3: *Distance [m] of the auxiliary path for each connection method for the different missions and map types.*

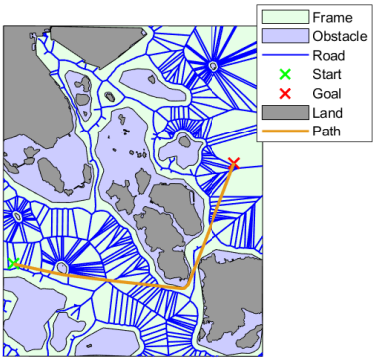
Map Type		WEO			NEO		
Method		CC	CNE	CCR	CC	CNE	CCR
Mission	1	47.78	47.78	47.78	47.78	47.78	47.78
	2	111.08	111.08	111.08	111.08	111.08	111.08
	3	83.82	83.82	84.59	121.01	83.81	84.59
	4	129.00	128.99	129.00	149.87	149.90	129.00
	5	26.21	26.21	26.21	37.12	26.23	26.23
	6	110.20	110.21	110.21	110.21	110.21	110.21
	7	40.46	40.45	40.46	40.47	40.45	40.45
	8	110.04	110.03	110.03	103.72	110.04	103.72
	9	74.61	74.61	74.61	184.06	184.04	74.61
	10	83.58	83.60	83.58	83.59	83.58	83.60

From Table 5.3 it can be seen that the connecting method does not matter much in the WEO map while it varies drastically in the NEO map. This is probably due to the density of edges in the WEO map and that it has more edges around obstacles that are not completely surrounded by other obstacles. The reason for the CC and CNE method performing worse in the NEO map is that there are no connections around some obstacles as can be seen in Figure 5.10. An exception to this is seen in Mission 8 where CC performs better on the NEO map. This is probably caused by the remove redundancies algorithm, described in Section 5.3.2, since CC and CNE have similar initial paths as can be seen in Figure 5.11. Connecting method CCR however, gives almost the same distance no matter the map type, with the exception of Mission 8. The exception in Mission 8 can be explained by the same reasoning as for the CC and CNE method for the same map, see Figure 5.12.

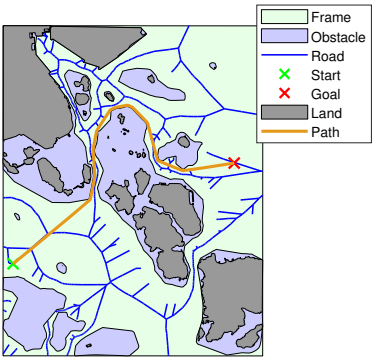
Small distance differences across the connecting methods are probably a result of the initial path from the graph search not being exactly the same due to the different connecting methods. Thus the smoothened path will not be either, even though it would if the maximum resolution in the smoothing algorithm was set to infinity. Hence, these lengths are considered to be equal.

Due to the lower computational complexity, the CC method is the best choice if the roadmap contains roads along the obstacles and thus creating road connections that otherwise would not be there. However, if there are no roads around the obstacles, the CCR method is probably a better choice than CC since it can obtain a path that is less than half the distance of the path acquired by the CC method.

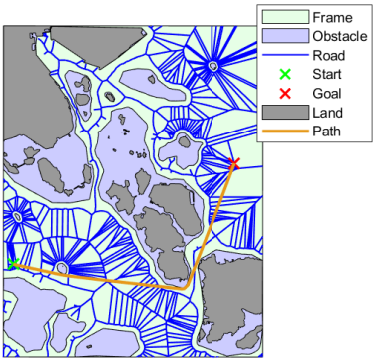
Another important aspect is the methods' ability to connect a position to the roadmap. The CCR method, for example, cannot connect a position to the roadmap if there are no crossroads in the map. However, this is unlikely to happen.



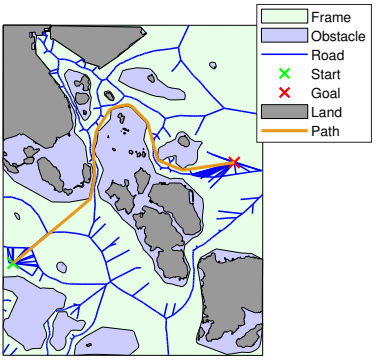
(a) Connecting method CC, map WEO.



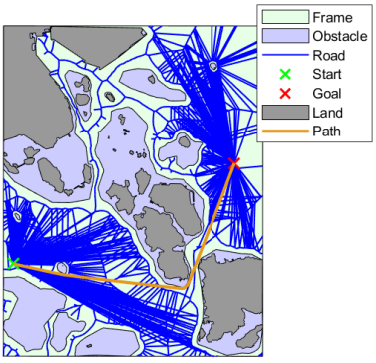
(b) Connecting method CC, map NEO.



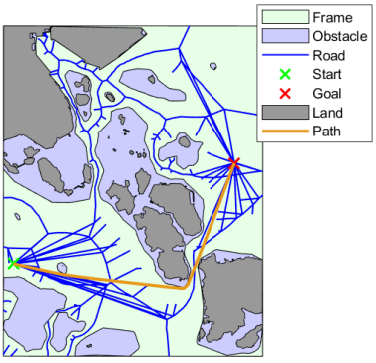
(c) Connecting method CNE, map WEO.



(d) Connecting method CNE, map NEO.

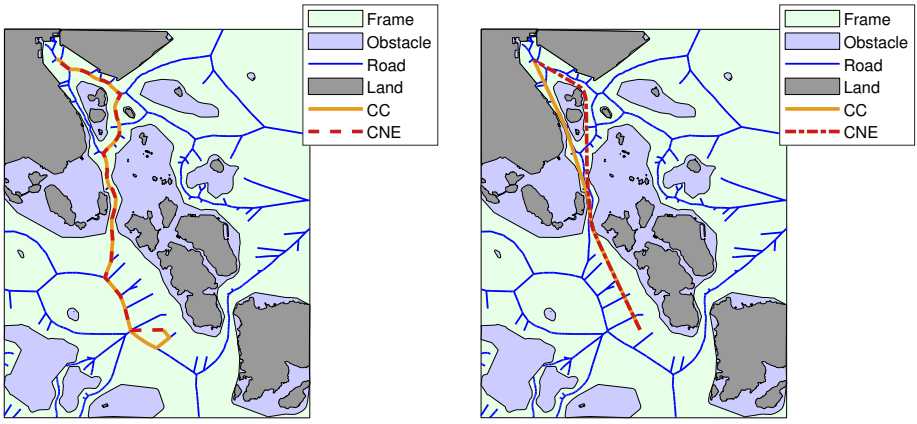


(e) Connecting method CCR, map WEO.



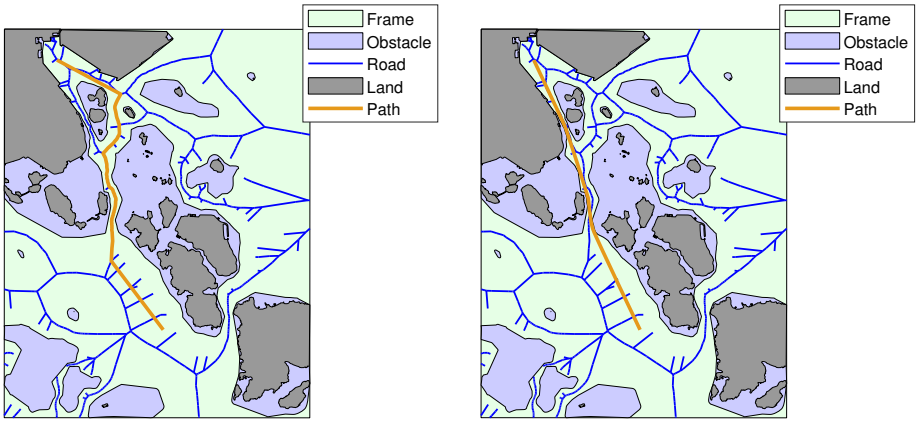
(f) Connecting method CCR, map NEO.

Figure 5.10: *Example of the use of different connection methods on the different maps for Mission 4.*



(a) *(b)*

Figure 5.11: *NEO roadmaps with the initial paths for Mission 8 (a) and the auxiliary path for Mission 8 (b).*



(a) *(b)*

Figure 5.12: *NEO roadmaps with the initial path for the CCR method for Mission 8 (a) and the auxiliary path for Mission 8 (b).*

Table 5.4: *Time [s] for path improvement.*

Mission	Remove redundancies	Smoothing
1	0.19	0
2	0.79	1.09
3	0.66	3.13
4	0.77	2.07
5	0.26	0.43
6	0.62	3.84
7	0.24	1.99
8	0.99	2.64
9	0.44	0
10	0.63	3.58

Table 5.5: *Number of vertices in the path when redundancies have been removed.*

Mission	1	2	3	4	5	6	7	8	9	10
Vertices	2	5	8	6	3	8	5	7	2	8

5.4.3 Path Smoothing

The minimum resolution Δ_{min} in Algorithm 5.2 is set to 5/16 and two incident edges were considered parallel if the angle between them was less than 0.57°.

A comparison of the computation time of path improvement between only removing redundancies and removing redundancies in combination with smoothing is presented in Table 5.4. The comparison is done over ten missions (the same ones as before) with connecting method CNE and map type WEO.

Table 5.4 and Table 5.3 suggest that the length of the path does not affect the computation time of the smoothing. Especially since the time of the improvement of Mission 3 is significantly longer than the time for Mission 2 even though the path length for Mission 2 is about 30% longer than Mission 3.

One reason for the smoothing taking longer time on some of the shorter paths than the longer ones could be caused by the number of curves needed to be smoothened since that would require more vertices to be checked if an improvement is possible. This is also strengthened by Table 5.5 when compared to Table 5.4 since a greater number of vertices in the path result in a longer computation time for the smoothing algorithm.

5.4.4 Conclusion

Since the mean computational time for creating a roadmap using generator points on expanded obstacles (WEO) is almost twice the time it takes to create a roudmap without generator points on expanded obstacles (NEO) and since the connecting to crossroads (CCR) method performs equally well on both types of maps in terms of distance, NEO combined with CCR is a good candidate. The computational time for CCR is also relatively short, thus the best combination of map type and connecting method is NEO and CCR. This combination is both fast and gives a short auxiliary path.

However, if any of the other connecting methods are used a WEO map could be used to decrease the path length. If the WEO map is used, the best connecting method would be the connecting to closest point (CC) since it is by far the fastest method and have similar performance in terms of length.

6

Trajectory Planning

The purpose of trajectory planning is to find a route that satisfies the dynamics of a vessel. The trajectory planning is based on a two-step approach where the first step is to calculate a simple initial trajectory and the second step is to optimize said trajectory. In this chapter, an initial trajectory is first generated from a geometric path, see Section 6.1. Then in Section 6.2, an optimal control problem is formed to improve the initial trajectory such that it satisfies the dynamics of the vessel. In Section 6.3, the results from the trajectory planning are presented as well as a short conclusion.

6.1 Initial Trajectory

The initial trajectory is obtained through a series of steps. First, a geometric path is obtained as described in Section 5.3. Then, a straight line trajectory (SLT) is calculated. The SLT is a trajectory created under the assumption that the vessel only moves in one direction. Lastly, a mapping of the SLT to a geometric path is performed to get a trajectory in the horizontal plane consisting of generalized position.

6.1.1 Straight Line Trajectory (SLT)

The SLT is created under the assumption that the vessel is travelling on a straight line and that it is subject to a maximum allowed velocity $V_{\max}$ and acceleration $a_{\max}$. A method for obtaining the SLT is called the linear segment with parabolic blends (LSPB) approach (Spong and Hutchinson, 2005). The LSPB approach is useful when it is desired to have a constant velocity over a portion of the trajectory (Spong and Hutchinson, 2005), and since there is a maximum allowed

velocity for the vessel this is useful in this thesis. The SLT $q(t)$ is split into five parts over different time intervals

$$q(t) = \begin{cases} q_1(0), & t < 0 \\ q_1(t), & 0 \leq t \leq t_b \\ q_2(t), & t_b < t \leq t_f - t_b \\ q_3(t), & t_f - t_b < t \leq t_f \\ q_3(t_f), & t_f < t \end{cases} \tag{6.1}$$

where $q_1(t)$ is the transient at the start where the vessel accelerates, $q_2(t)$ is when the vessel is moving with constant velocity along the path and $q_3(t)$ is the deceleration such that the vessel comes to a halt (Spong and Hutchinson, 2005). The time t_b of the switch from $q_1(t)$ to $q_2(t)$ is called blending time and t_f is the total time of the route (Spong and Hutchinson, 2005).

The transient $q_1(t)$ at the start is subject to

$$q_1(t_0) = q_0 = 0 \tag{6.2a}$$
$$\dot{q}_1(t_0) = V_0 = 0 \tag{6.2b}$$
$$\ddot{q}_1(t_0) = a_0 = 0 \tag{6.2c}$$
$$\dot{q}_1(t_b) = V_{\max} \geq 0 \tag{6.2d}$$
$$\ddot{q}_1(t_b) = a_{t_b} = 0 \tag{6.2e}$$
$$|\ddot{q}_1(t)| \leq a_{\max}, \quad t_0 \leq t \leq t_b \tag{6.2f}$$

with $t_0 = 0$. In words, the transient is starting at the position q_0 with the velocity V_0 and acceleration a_0 at time t_0, then accelerates to a maximum and constant velocity $V_{\max}$ at time t_b without exceeding the maximum allowed acceleration $a_{\max}$. Since $q_1(t)$ has six constraints and t_b is a free variable $q_1(t)$ needs to be a forth degree polynomial

$$q_1(t) = b_{10} + b_{11}t + b_{12}t^2 + b_{13}t^3 + b_{14}t^4.$$

Since $\ddot{q}_1(t)$ is a second order polynomial and $\dot{q}_1(t_b) \geq 0$ this implies that $\ddot{q}_1(t) \geq 0$ on the interval $t \in [t_0,\ t_b]$. Thus, the constraint (6.2f) can be rewritten as

$$0 \leq \ddot{q}_1(t) \leq a_{\max}, \quad 0 \leq t \leq t_b. \tag{6.3}$$

Since it is desired to minimize the time of the trajectory, let

$$\ddot{q}_1(t_m) = a_{\max}, \quad 0 < t_m < t_b, \tag{6.4}$$

where $t = t_m$ is at the maximum of $\ddot{q}_1(t)$ for $t \in (0,\ t_b)$. Then, $V_{\max}$ will be achieved as fast as possible.

Solving the system of equations (6.2a–e) given the constraints (6.3) and (6.4) results in

$$b_{10} = 0, \quad b_{11} = 0, \quad b_{12} = 0, \quad b_{13} = \frac{4a_{\max}^2}{9V_{\max}}, \quad b_{14} = -\frac{4a_{\max}^3}{27V_{\max}^2} \tag{6.5}$$

and

$$t_b = \frac{3V_{\max}}{2a_{\max}}$$

thus

$$q_1(t) = \frac{4a_{\max}^2}{9V_{\max}} t^3 \left(1 - \frac{a_{\max}}{3V_{\max}} t\right). \tag{6.6}$$

Trajectory $q_2(t)$ is a linear function according to

$$q_2(t) = q(t_b) + V_{\max}(t - t_b) \tag{6.7}$$

(Spong and Hutchinson, 2005) and the third segment of the trajectory is

$$q_3(t) = b_{30} + b_{31}t + b_{32}t^2 + b_{33}t^3 + b_{34}t^4$$

with the constraints

$$\begin{aligned}
\dot{q}_3(t_f - t_b) &= V_{\max} \geq 0 \\
\ddot{q}_3(t_f - t_b) &= 0 \\
q_3(t_f) &= q_f \\
\dot{q}_3(t_f) &= 0 \\
\ddot{q}_3(t_f) &= 0 \\
-a_{\max} \leq \ddot{q}_3(t) &\leq 0, \quad t_f - t_b < t \leq t_f
\end{aligned}$$

where q_f is the total distance of the trajectory. Due to symmetry

$$q_3(t) = q_f - q_1(t_f - t) \tag{6.8}$$

(Spong and Hutchinson, 2005) and with q_f known, the time t_f can be calculated using

$$t_f = \frac{q_f - 2q(t_b) + 2t_b V_{\max}}{V_{\max}}.$$

In the case where

$$t_b = \frac{3V_{\max}}{2a_{\max}} > \frac{t_f}{2}$$

the trajectory described above is infeasible, since there is not sufficient time to reach the maximum velocity. To solve this issue, a new blend time $t_{b,2}$ is calculated. The time $t_{b,2}$ is set to when half the total distance is traversed

$$\frac{q_f}{2} = q(t_{b,2}) = \frac{4a_{\max}^2}{9V_{\text{new}}} t_{b,2}^3 \left(1 - \frac{a_{\max}}{3V_{\text{new}}} t_{b,2}\right) \tag{6.9}$$

where V_{new} is a new maximum allowed velocity. The new maximum velocity can, due to symmetry be written as

$$V_{\text{new}} = \frac{q_f}{t_{b,2}}. \tag{6.10}$$

Solving (6.9) for $t_{b,2}$ using (6.10) gives

$$t_{b,2} = \sqrt{\frac{3q_f}{2a_{\max}}}$$

then the new final time will be

$$t_{f,2} = 2t_{b,2}.$$

By replacing $V_{\max}$ with (6.10) in (6.5), a feasible trajectory is acquired. The trajectory will accelerate the first half and achieve a velocity of V_{new} at time $t_{b,2} = \frac{t_{f,2}}{2}$. The second half will be a deceleration to rest at time $t_{f,2}$.

6.1.2 Mapping SLT to Geometric Path

When the SLT is mapped to the geometric path, a discrete trajectory in the plane describing the generalized position of the vessel is acquired. The trajectory also contains a surge velocity $u(t) = \dot{q}(t)$.

In order to map the SLT to a geometric path, the SLT is first sampled at time intervals T_s such that, $q[n] = q(nT_s)$ for $n = 0, 1, \ldots, N$, where $N = \left\lceil \frac{t_f}{T_s} \right\rceil$. By letting the distance d_i for the i:th vertex $\boldsymbol{v}_i$ in the geometric path be the cumulative distance from all the previous vertices including the distance from vertex $\boldsymbol{v}_{i-1}$ to $\boldsymbol{v}_i$. Then, for each sample $q[n]$, the previous and next vertex on the geometric path is found. The previous vertex $\boldsymbol{v}_{\text{prev}}$ is the vertex with distance closest to $q[n]$ and satisfying $d_i \leq q[n]$, for some $n = 0, 1, \ldots, N$, and the next vertex $\boldsymbol{v}_{\text{next}}$ is the vertex on the geometric path closest to $q[n]$ where $q[n] < d_i$. With $\boldsymbol{V}_{\text{dir}}$ being a vector with length one, pointing from $\boldsymbol{v}_{\text{prev}}$ to $\boldsymbol{v}_{\text{next}}$, then the position $\boldsymbol{p}_n$ of the n:th sample is given by

$$\boldsymbol{p}_n = (q[n] - d_{\text{prev}})\boldsymbol{V}_{\text{dir}} + \boldsymbol{v}_{\text{prev}} \tag{6.11}$$

where d_{prev} is the distance to $\boldsymbol{v}_{\text{prev}}$. If there is no $d_i > q[n]$, then $\boldsymbol{v}_{\text{prev}}$ is set to be the second to last vertex and $\boldsymbol{v}_{\text{next}}$ is set to be the last vertex in the geometric path and $\boldsymbol{p}_n = \boldsymbol{v}_{\text{next}}$ instead of (6.11).

The orientation ψ_n of the vessel at sample n is given by

$$\psi_n = \text{atan2}(\boldsymbol{V}_{\text{dir},y}, \boldsymbol{V}_{\text{dir},x}) \tag{6.12}$$

where $\boldsymbol{V}_{\text{dir},x}$ and $\boldsymbol{V}_{\text{dir},y}$ are the x and y component of $\boldsymbol{V}_{\text{dir}}$ respectively. This procedure will ensure that the initial trajectory is within the bounds of the free-space $\mathcal{X}_{\text{free}}$ since the auxiliary path described in Section 5.3 is. The algorithm is presented in pseudo code in Algorithm 6.1.

Algorithm 6.1: Mapping SLT to geometric path

Input: A path $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k\}$, a trajectory $q(t)$, a sample time T_s and the total time t_f
Output: A discrete trajectory in the plane $\mathcal{T}$

$\mathcal{T} \longleftarrow \emptyset$
$N \longleftarrow \left\lceil \frac{t_f}{T_s} \right\rceil$
for $n = 0$ **to** N **do**
  $q_n \longleftarrow q(nT_s)$
  $u_n \longleftarrow \dot{q}(nT_s)$ `// Surge velocity`
  $j_{\text{prev}} \longleftarrow$ index of $\boldsymbol{v} \in \mathcal{V}$ with distance d closest to q_n satisfying $d \leq q_n$
  $j_{\text{next}} \longleftarrow$ index of $\boldsymbol{v} \in \mathcal{V}$ with distance d closest to q_n satisfying $d > q_n$
  if j_{next} *does* **not** *exist* **then**
    $\boldsymbol{v}_{\text{prev}} \longleftarrow \boldsymbol{v}_{j_{\text{prev}}-1}$
    $\boldsymbol{v}_{\text{next}} \longleftarrow \boldsymbol{v}_{j_{\text{prev}}}$
    $\boldsymbol{V}_{\text{dir}} \longleftarrow$ the normalized vector going from $\boldsymbol{v}_{\text{prev}}$ to $\boldsymbol{v}_{\text{next}}$
    $\boldsymbol{p}_n \longleftarrow \boldsymbol{v}_{\text{next}}$
  else
    $\boldsymbol{v}_{\text{prev}} \longleftarrow \boldsymbol{v}_{j_{\text{prev}}}$
    $\boldsymbol{v}_{\text{next}} \longleftarrow \boldsymbol{v}_{j_{\text{next}}}$
    $\boldsymbol{V}_{\text{dir}} \longleftarrow$ the normalized vector going from $\boldsymbol{v}_{\text{prev}}$ to $\boldsymbol{v}_{\text{next}}$
    $\boldsymbol{p}_n \longleftarrow (q_n - d_{\text{prev}})\boldsymbol{V}_{\text{dir}} + \boldsymbol{v}_{\text{prev}}$
  end
  $\psi_n \longleftarrow \text{atan2}(\boldsymbol{V}_{\text{dir},y}, \boldsymbol{V}_{\text{dir},x})$
  $\mathcal{T} \longleftarrow$ insert position $\boldsymbol{p}_n$, heading ψ_n and surge u_n at time nT_s
end

6.2 Trajectory Optimization

To make the trajectory dynamically feasible for the vessel, an OCP is formed and solved using multiple shooting, as described in Chapter 3.

With the state vector $\boldsymbol{x}_e = \begin{bmatrix} \boldsymbol{x}^T & \boldsymbol{u}^T \end{bmatrix}^T$ and with $\boldsymbol{u}_e = \dot{\boldsymbol{u}}$, where $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{\vartheta}$ are the same as in Section 2.3, the state space model becomes

$$\dot{\boldsymbol{x}}_e = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\vartheta}) \\ \boldsymbol{u}_e \end{bmatrix} \tag{6.13}$$

where $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\vartheta})$ is given by (2.24a). The reason why $\boldsymbol{u}$ is included in the state vector instead of being treated as a control input is to take the dynamics of the thrusters into account (Bergman et al., 2020).

6.2.1 Collision Avoidance

An efficient representation of the free-space is essential in order to reduce the computational complexity of the trajectory optimization step. One method described by Bergman et al. (2020), is to compute local spatial constraints for each sample on the initial trajectory. The constraint for sample i on the initial trajectory is represented by a convex polytope $\mathcal{S}^i_{\text{env}}$ defined by a number of half planes K_i such that

$$\mathcal{S}^i_{\text{env}} = \{ \boldsymbol{p} \in \mathbb{R}^2 \mid \boldsymbol{A}_i \boldsymbol{p} \leq \boldsymbol{b}_i \} \tag{6.14}$$

where $\boldsymbol{A}_i = \begin{bmatrix} \boldsymbol{a}_{i,1} & \cdots & \boldsymbol{a}_{i,K_i} \end{bmatrix}^T \in \mathbb{R}^{K_i \times 2}$, $\boldsymbol{a}_{i,j} \in \mathbb{R}^2$ for $j = 1, \ldots, K_i$, and $\boldsymbol{b}_i \in \mathbb{R}^{K_i}$ (Bergman et al., 2020). The approach is based on the expansion of a convex polytope $\mathcal{S}_b$ approximating the body of the vessel. The algorithm used to compute $\mathcal{S}^i_{\text{env}}$ in this report is the same as the one described by Bergman et al. (2020), but with a few modifications for easier implementation.

As in Bergman et al. (2020), first $\mathcal{S}^i_{\text{env}}$ is initialized as a directed cyclic graph $\mathcal{G}^i_{\text{env}} =< \mathcal{V}^i_{\text{env}}, \mathcal{E}^i_{\text{env}} >$ where $\boldsymbol{p}_k \in \mathcal{V}^i_{\text{env}}$ composes the convex hull of $\mathcal{S}^i_{\text{env}}$ and the edge $e_k \in \mathcal{E}^i_{\text{env}}$ represents a vector from $\boldsymbol{p}_k$ to $\boldsymbol{p}_{k+1}$. For each $\boldsymbol{p}_k$ there is an associated expansion direction $\boldsymbol{g}_k$, selected such that the area spanned by the graph $\mathcal{G}^i_{\text{env}}$ is increased after expansion. An example of an expansion direction that was used in this report is given by Bergman et al. (2020), which is to choose $\boldsymbol{g}_k$ to be parallel to the vector going from the center of the ship's body to the initial vertex $\boldsymbol{p}_k$.

The first part (Line 1 - Line 7) of Algorithm 6.2 is exactly the same as in Bergman et al. (2020). The graph $\mathcal{G}^i_{\text{env}}$ is initialized using $\mathcal{S}_b$ at $\boldsymbol{\eta}_i$, see Figure 6.1(a). The initial vertex $\boldsymbol{p}_c$ is then selected as the vertex with expansion direction closest to $\boldsymbol{\nu}_i$. Then, the graph is expanded for as long as any of the vertices are expandable. Vertex $\boldsymbol{p} \in \mathcal{V}^i_{\text{env}}$ is not considered to be expandable if the corresponding step length Δ_l falls below a minimum value or the expansion reaches a maximum distance. If $\boldsymbol{p}$ is expandable, it is expanded in its direction $\boldsymbol{g}$ with step length Δ_l.

The second part is where the change has been done (Line 8 - Line 16). After a vertex is expanded it is checked if the shape of the graph is still convex. If the graph is not convex, then the expansion is reverted and the corresponding step length Δ_l is divided by two and the next vertex in the graph is processed, see Figure 6.1(c) and Figure 6.1(d). If the graph is convex, then, if any of the edges $\mathcal{E}^i_{\text{env}}$ intersect with $\mathcal{E}_{\text{obst}}$, the expansion is dismissed and the step length Δ_l is halved, see Figure 6.1(b) and Figure 6.1(c), then the next vertex is processed.

Algorithm 6.2: Computation of local spatial constraints

Input: $\boldsymbol{\eta}_i$, $\boldsymbol{\nu}_i$, $\mathcal{S}_b$, $\mathcal{E}_{\text{obst}}$
Output: $\mathcal{S}^i_{\text{env}}$

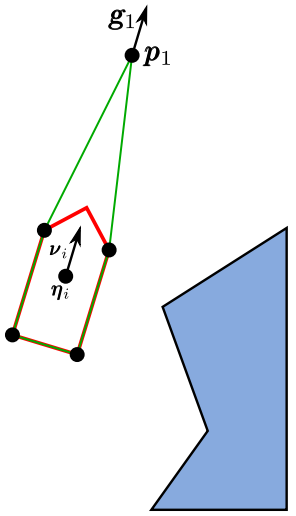
1 $\mathcal{G}^i_{\text{env}} =< \mathcal{V}^i_{\text{env}}, \mathcal{E}^i_{\text{env}} >\longleftarrow$ a cyclic graph of $\mathcal{S}_b$ given $\boldsymbol{\eta}_i$
2 $\boldsymbol{p}_c \longleftarrow$ vertex $\boldsymbol{p} \in \mathcal{V}^i_{\text{env}}$ with expansion direction closest to $\boldsymbol{\nu}_i$
3 **while** $\mathcal{G}^i_{env}$ *is expandable* **do**
4 **if** $\boldsymbol{p}_c$ *is expandable* **then**
5 $\Delta_l \longleftarrow$ step length of $\boldsymbol{p}_c$
6 $\boldsymbol{g} \longleftarrow$ expansion direction of $\boldsymbol{p}_c$
7 $\boldsymbol{p}_c \longleftarrow \boldsymbol{p}_c + \Delta_l \boldsymbol{g}$
8 **if** *the shape of* $\mathcal{G}^i_{env}$ *is* **not** *convex* **then**
9 $\boldsymbol{p}_c \longleftarrow \boldsymbol{p}_c - \Delta_l \boldsymbol{g}$
10 step length of $\boldsymbol{p}_c \longleftarrow \Delta_l/2$
11 **else**
12 **if** *any of the edges* $\mathcal{E}^i_{env}$ *intersect* $\mathcal{E}_{obst}$ **then**
13 $\boldsymbol{p}_c \longleftarrow \boldsymbol{p}_c - \Delta_l \boldsymbol{g}$
14 step length of $\boldsymbol{p}_c \longleftarrow \Delta_l/2$
15 **end**
16 **end**
17 **end**
18 $\boldsymbol{p}_c \longleftarrow$ next vertex in $\mathcal{V}^i_{\text{env}}$
19 **end**
20 $\mathcal{S}^i_{\text{env}} \longleftarrow$ span of $\mathcal{V}^i_{\text{env}}$

The condition for collision avoidance is then

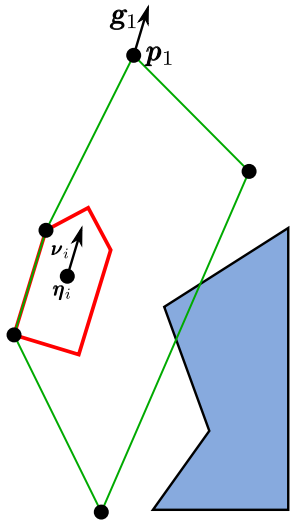
$$\boldsymbol{A}_i \boldsymbol{c}_{\text{rot}}(\boldsymbol{x}_{e,i}, \boldsymbol{p}_j) \leq \boldsymbol{b}_i, \quad \forall \boldsymbol{p}_j \in \mathcal{V}_b \tag{6.15}$$

where $\mathcal{V}_b$ are the vertices of $\mathcal{S}_b$, $\boldsymbol{A}_i$ and $\boldsymbol{b}_i$ are the half-plane representation of $\mathcal{S}^i_{\text{env}}$ in (6.14) and

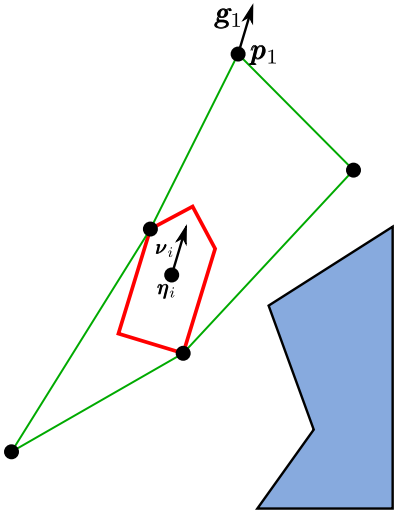
$$\boldsymbol{c}_{\text{rot}}(\boldsymbol{x}_{e,i}, \boldsymbol{p}_j) = \begin{bmatrix} \cos\psi_i & -\sin\psi_i \\ \sin\psi_i & \cos\psi_i \end{bmatrix} \boldsymbol{p}_j + \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{6.16}$$



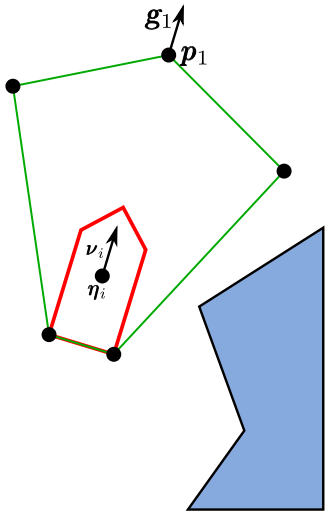
(a) Initialize $\mathcal{G}^i_{env}$ using $\mathcal{S}_b(\boldsymbol{\eta}_i)$ and expand in direction closest to $\boldsymbol{\nu}_i$.



(b) Continue expanding $\boldsymbol{p} \in \mathcal{V}^i_{env}$ that are expandable. Intersection detected, revert last expansion and halve the step length for that $\boldsymbol{p}$.



(c) Span of $\mathcal{V}^i_{env}$ not convex, revert last expansion and halve step length for that $\boldsymbol{p}$.



(d) $\mathcal{G}^i_{env}$ after one cycle of expansions.

Figure 6.1: *Expanding $\mathcal{S}_b(\boldsymbol{\eta}_i)$ (red) to form the spatial constraints for collision avoidance. In the subfigures, the graph $\mathcal{G}^i_{env}$ is green, obstacle are blue and the vertices $\boldsymbol{p} \in \mathcal{V}^i_{env}$ are black circles.*

6.2.2 Problem Formulation

The discrete OCP can be formulated as in Bergman et al. (2020)

$$\underset{\{\boldsymbol{u}_{e,i}\}_{i=0}^{N-1},\Delta_t}{\text{minimize}}\ J_d = \sum_{i=0}^{N-1} \mathcal{L}(\boldsymbol{x}_{e,i}, \boldsymbol{u}_{e,i}, \Delta_t) \tag{6.17a}$$

$$\text{s.t. } \boldsymbol{x}_{e,0} = \boldsymbol{x}_{e,\text{cur}}, \quad \boldsymbol{b}_{l,N} \leq \boldsymbol{x}_{e,N} \leq \boldsymbol{b}_{u,N} \tag{6.17b}$$

$$\boldsymbol{x}_{e,i+1} = \boldsymbol{F}(\boldsymbol{x}_{e,i}, \boldsymbol{u}_{e,i}, \Delta_t) \tag{6.17c}$$

$$\boldsymbol{A}_i \boldsymbol{c}_{\text{rot}}(\boldsymbol{x}_{e,i}, \boldsymbol{p}_j) \leq \boldsymbol{b}_i, \ \forall \boldsymbol{p}_j \in \mathcal{V}_b \tag{6.17d}$$

$$\boldsymbol{x}_{e,i} \in \mathcal{X}_{\text{free}}, \quad \boldsymbol{u}_{e,i} \in \mathcal{U} \tag{6.17e}$$

The decision variable Δ_t represents the time between two consecutive samples. The function $\mathcal{L}$ is the numerical integral of

$$\ell(\boldsymbol{x}_e, \boldsymbol{u}_e) = \frac{10^4}{T_h} + u^2 \boldsymbol{\alpha}^T \boldsymbol{\alpha} + 10^{-1} \boldsymbol{n}^T \boldsymbol{n} + 10^2 \dot{\boldsymbol{\alpha}}^T \dot{\boldsymbol{\alpha}} + 10^2 \dot{\boldsymbol{n}}^T \dot{\boldsymbol{n}} \tag{6.18}$$

over the time Δ_t, where T_h is the planning horizon time, $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix}^T$ and $\boldsymbol{n} = \begin{bmatrix} n_1 & n_2 & n_t \end{bmatrix}^T$. The function $\boldsymbol{F}$ is the state-space model (6.13), discretized using the RK4 method. For the final state $\boldsymbol{x}_{e,N}$, there is a lower and upper bound, denoted $\boldsymbol{b}_{l,N}$ and $\boldsymbol{b}_{u,N}$, respectively. This freedom is introduced since the initial trajectory may not be achievable. Both $\boldsymbol{A}_i$ and $\boldsymbol{b}_i$ are the half-plane representation at the sample point on the initial trajectory that is closest to $\boldsymbol{x}_{e,i}$. Finally, $\boldsymbol{x}_{e,\text{cur}}$ is the sample on the trajectory that the optimization starts from.

The lower and upper bounds are given by

$$\boldsymbol{b}_{l,N} = \begin{bmatrix} x_{e,\text{cur}+N} & y_{e,\text{cur}+N} & \psi_{e,\text{cur}+N} & u_{e,\text{cur}+N} & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty \end{bmatrix}^T \tag{6.19a}$$

$$\boldsymbol{b}_{u,N} = \begin{bmatrix} x_{e,\text{cur}+N} & y_{e,\text{cur}+N} & \psi_{e,\text{cur}+N} & u_{e,\text{cur}+N} & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}^T \tag{6.19b}$$

where $\boldsymbol{x}_{e,\text{cur}+N}$ is the sample on the trajectory N steps ahead of $\boldsymbol{x}_{e,\text{cur}}$.

The problem can then be solved for the whole initial trajectory at once or repeatedly using a receding horizon approach (Bergman et al., 2020). An advantage to the receding horizon approach is its ability to handle unforeseen obstacles since it updates a section of the initial trajectory at a time. But since the environment is considered static in this thesis, that advantage is lost.

6.2.3 Receding Horizon

For the receding horizon approach, the trajectory is optimized a short segment at a time. The segment to be optimized stretches over a time period of T_h. After the segment is optimized, a new segment starting from the next sample on the trajectory and ending N samples ahead of that one, see Figure 6.2. The receding horizon approach also results in a suboptimal trajectory (Xu et al., 2010).

In order to maintain a constant planning horizon T_h, the part of the trajectory in Figure 6.2(a) that was optimized and the part that still has to be optimized need

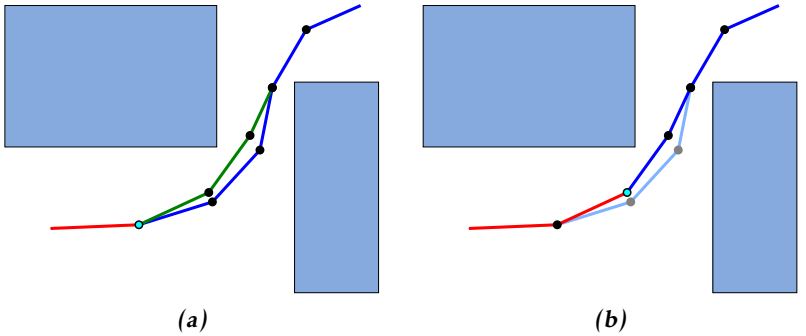
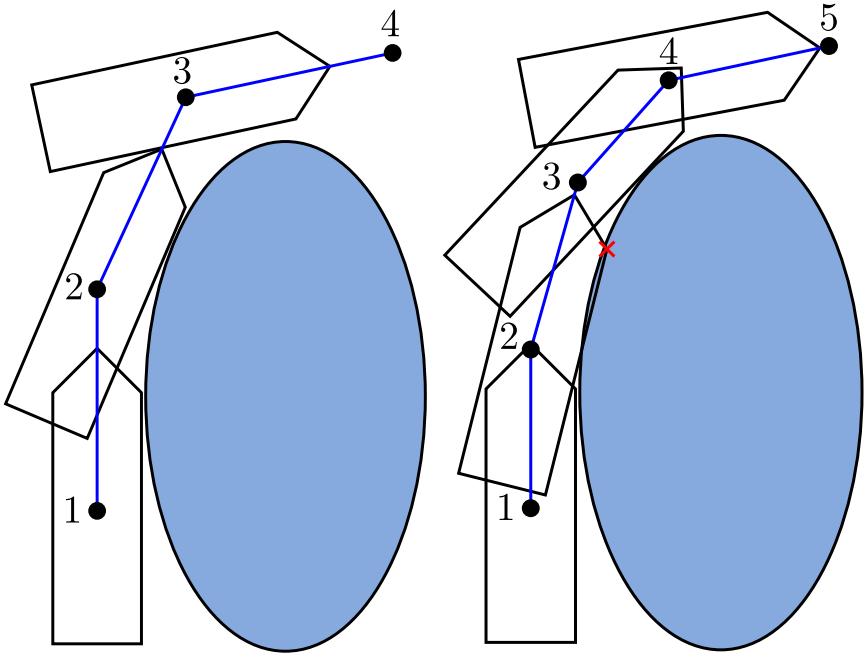


Figure 6.2: *Example of receding horizon approach. Blue squares are obstacles, blue lines corresponds to the initial trajectory, red lines corresponds to the part of the trajectory that has been optimized in previous steps, green line is the segment that is currently being optimized and light blue line is the trajectory before optimization. The black circles represent the samples of the trajectory and the cyan dot marks the first sample on the horizon. (a), Optimize the trajectory over, e.g. three samples, starting at the cyan circle ($\boldsymbol{x}_{e,cur}$). (b), Replace the old trajectory with the optimized and move on to the next sample.*

to be resampled. This is a consequence of Δ_t being a decision variable which means that Δ_t can vary over different segments if a receding horizon approach is used. Moving one sample forward would not always correspond to a time step of T_s as in the initial trajectory. Thus, if $\Delta_t < T_s$, then T_h would decrease and increase if $\Delta_t > T_s$.

However, a problem with resampling is that the new samples may not satisfy the OCP (6.17). For instance, if one of the vertices in $\mathcal{S}_b$ does not satisfy (6.17d) for the sample that would be $\boldsymbol{x}_{e,0}$ in the next iteration, then (6.17) has no solution, see Figure 6.3. If sample 1 is the current initial sample, the spatial constraints (6.17d) cannot be satisfied for $i = 0$ when resampling and the next sample (sample 2) is set to the current, since $\boldsymbol{x}_{e,0}$ is fixed. This problem is avoided by not performing resampling and letting the planning horizon change. It can also be avoided by optimizing the whole initial trajectory at once.



(a) *Before resampling.* ***(b)*** *After resampling.*

Figure 6.3: *Trajectory (blue), $\mathcal{S}_b(\boldsymbol{x}_{e,i})$ (black), samples (black circles) and obstacle (light blue) for the resampling. Red cross is where vertex in $\mathcal{S}_b(\boldsymbol{x}_{e,i})$ is inside the obstacle.*

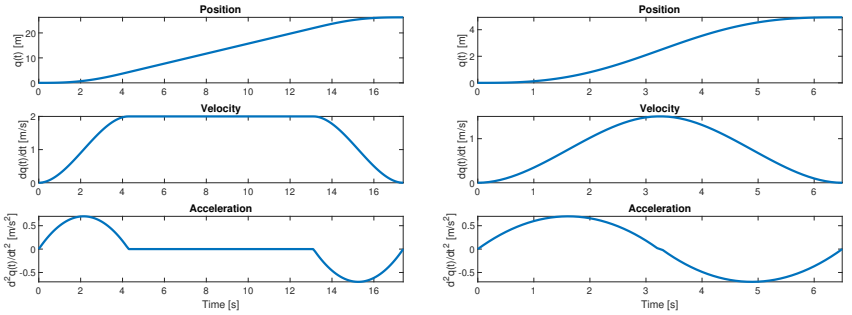


Figure 6.4: SLT for Mission 5 (a) and SLT with adjusted V_{max} due to short distance (b).

6.3 Trajectory Planning Result

The parameters used for acquiring the auxiliary path are the same as in Section 5.4. Two different ship models were examined, one of which is the model estimated in Section 4.5.2 (estimated model) and the other is given in Section 4.5.1 but with $\kappa = 0$, no wind forces and without the tunnel thruster (simulated model).

6.3.1 SLT and Mapping to Geometric Path

The SLT is generated with $V_{\max} = 2$ and $a_{\max} = 0.7$ and the sample time $T_s = 0.1$ seconds for the simulated model. For the estimated model $V_{\max} = 1$, $a_{\max} = 1$ and $T_s = 0.1$. The SLT of Mission 5 for the simulated model can be seen in Figure 6.4(a) and the corresponding mapping to geometric path is shown in Figure 6.5

6.3.2 Optimization

For the estimated model, the planning horizon T_h is set to 30 seconds and the number of control intervals is set to $N = 75$ for the receding horizon approach and when optimizing the whole trajectory at once each control interval spans over 0.4 seconds for the estimated model. The set of control signals is $\mathcal{U} = \{\boldsymbol{u}_e \mid \mathbf{lb}_u \leq \boldsymbol{u}_e \leq \mathbf{ub}_u\}$ and the states are limited by $\mathcal{X} = \{\boldsymbol{x}_e \mid \mathbf{lb}_x \leq \boldsymbol{x}_e \leq \mathbf{ub}_x\}$ where

$$\mathbf{lb}_u = -\begin{bmatrix} \frac{10}{9}\pi & \frac{10}{9}\pi & \frac{175}{2}\pi & \frac{175}{2}\pi & \frac{80}{3}\pi \end{bmatrix}^T$$

$$\mathbf{ub}_u = -\mathbf{lb}_u$$

$$\mathbf{lb}_x = -\begin{bmatrix} \infty & \infty & \infty & 2 & 2 & 1 & \pi & \pi & 0 & 0 & 140\pi \end{bmatrix}^T$$

$$\mathbf{ub}_x = \begin{bmatrix} \infty & \infty & \infty & 2 & 2 & 1 & \pi & \pi & 90\pi & 90\pi & 140\pi \end{bmatrix}^T$$

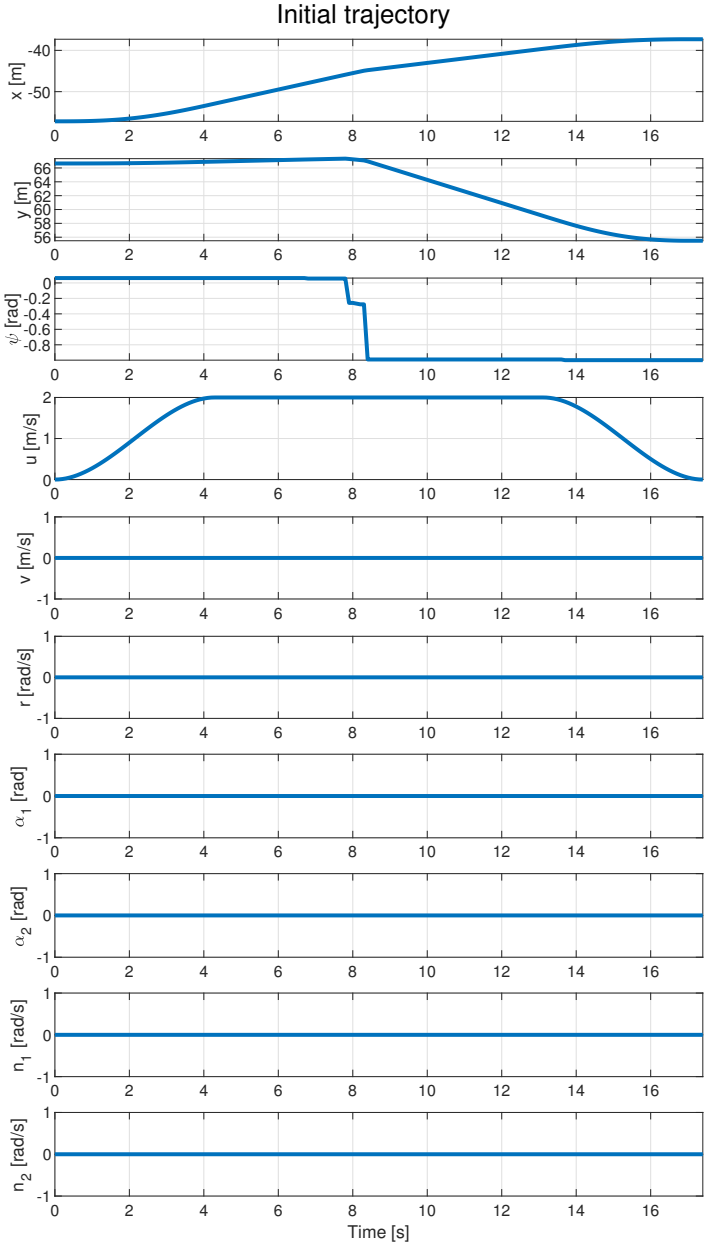


Figure 6.5: *SLT for Mission 5 mapped to the geometric path. The actuator signals* $\boldsymbol{u}$ *as well as the surge and sway velocities are set to 0 as an initial guess.*

Table 6.1: *Time duration [s] of the optimization (including time for path smoothing) for the simulated model with receding horizon and whole at once approaches, both initialized with and without smoothened initial trajectory (SIT and NSIT).*

Mission	Receding horizon		Whole at once	
	NSIT	SIT	NSIT	SIT
1	27.34	**24.25**	236.88	244.63
2	102.95	85.78	307.30	**22.63**
3	89.33	72.56	**18.42**	19.08
4	129.85	105.22	862.54	**41.44**
5	7.95	6.22	**5.63**	7.48
6	109.46	118.36	**42.72**	400.64
7	27.56	21.99	11.81	**10.45**
8	103.72	100.57	**18.27**	1580.92
9	51.46	52.40	**10.39**	10.93
10	85.16	**68.63**	273.61	212.40

For the simulated model, $T_h = 16$ seconds, $N = 40$ for the receding horizon and similarly to the estimated model, for the whole at once optimization approach each control interval spans over 0.4 seconds. The bounds are

$$\mathbf{lb}_u = -\begin{bmatrix}100 & 100 & 200 & 200 & 0\end{bmatrix}^T$$
$$\mathbf{ub}_u = -\mathbf{lb}_u$$
$$\mathbf{lb}_x = -\begin{bmatrix}\infty & \infty & \infty & 2 & 2 & 1 & \pi & \pi & 0 & 0 & 0\end{bmatrix}^T$$
$$\mathbf{ub}_x = \begin{bmatrix}\infty & \infty & \infty & 2 & 2 & 1 & \pi & \pi & 300 & 300 & 0\end{bmatrix}^T$$

The decision variable $\Delta_t \in [0,\, 2T_h/N]$ for receding horizon and $\Delta_t \in [0,\, 2T_f/N]$ otherwise, where T_f is the total time of the initial trajectory. For both models the collision avoidance constraints were calculated using $\Delta_l = 5$ initially, with minimum step length of 0.1 and maximum expansion distance 20.

Data are collected from the same ten missions as in Section 5.4.2 with the connecting method CNE and map type WEO for both receding horizon and the whole trajectory at once. Also data from optimizing the initial trajectory created from an auxiliary path that has not been smoothened, was examined. The optimization was done in MATLAB using the CasADi framework (Andersson et al., 2019) with IPOPT solver (Wächter and Biegler, 2005).

As can be seen in Table 6.1 and Table 6.2, optimizing the whole trajectory at once is better in terms of computational time with only a few of exceptions. For the simulated model, initializing with a non-smoothened trajectory is better while for the estimated model, initiating with a smoothened trajectory result in shorter computation time. The outliers could be a result from the initial guess (initial

Table 6.2: *Time duration [s] of the optimization (including time for path smoothing) for the estimated model with receding horizon and whole at once approaches, both initialized with and without smoothened initial trajectory (SIT and NSIT).*

Mission	Receding horizon		Whole at once	
	NSIT	SIT	NSIT	SIT
1	**54.10**	55.14	240.34	253.74
2	391.49	1113.02	2107.35	**234.54**
3	167.84	409.65	173.97	**43.48**
4	309.35	435.65	**38.60**	41.03
5	8.65	10.18	**8.12**	9.38
6	1370.12	2544.92	**55.44**	58.94
7	41.20	38.86	**6.92**	12.30
8	317.31	2521.31	**52.34**	137.27
9	135.67	135.26	**21.24**	21.32
10	170.86	1512.67	38.60	**23.82**

trajectory) being infeasible or far from optimal, thus struggling to minimize the objective function. It could also be a consequence of the collision avoidance constraints being too tight, leading to local infeasibility. However, to get a better understanding of this result a deeper knowledge of the IPOPT solver is needed.

The computation time of the receding horizon approach for the estimated model is not as consistent as for the simulated model. This could be a result from the planning horizon T_h being too short for the estimated model, thus converging to local infeasibility.

For the receding horizon approach, initializing the problem with an initial trajectory based on a smoothened path decreases the computational time for the simulated model, while it increases the computation time for the estimated model. Initializing the receding horizon approach with a trajectory based on smooth path seems very unreliable for the estimated model, see Table 6.2. For the whole at once approach, both models produces better results when the problem is initialized with the trajectory that is not based on a smooth path.

In Table 6.3 and Table 6.4, the time of the planned trajectory is presented and both models indicates that the receding horizon approach with smoothened initial trajectory gives a shortest time trajectory. However, both approaches and initialization methods result in times that are only a few seconds apart, at most.

Similarly to the time of the trajectory in Table 6.3 and Table 6.4, the distance of the trajectory presented in Table 6.5 and Table 6.6 are the shortest for the receding horizon approach initialized with a smooth initial trajectory. This holds for both models. However, the trajectory length only differs a few meters at most and since the ship's length is 0.99 meters the difference is not significant.

Table 6.3: *Total time [s] of the planned trajectory for the simulated model with the receding horizon and whole at once approaches, both initialized with and without smoothened initial trajectory (SIT and NSIT).*

Mission	Receding horizon		Whole at once	
	NSIT	SIT	NSIT	SIT
1	**27.35**	**27.35**	27.95	27.95
2	59.23	**59.05**	61.98	61.41
3	46.89	**45.10**	49.25	47.27
4	69.69	**67.73**	73.64	70.30
5	16.83	**16.27**	17.45	16.90
6	**59.29**	59.49	60.65	60.64
7	23.68	**23.46**	24.43	24.07
8	59.98	**58.37**	62.69	60.99
9	**40.76**	**40.76**	41.84	41.84
10	51.67	**44.78**	47.11	46.40

Table 6.4: *Total time [s] of the planned trajectory for the estimated model with the receding horizon and whole at once approaches, both initialized with and without smoothened initial trajectory (SIT and NSIT).*

Mission	Receding horizon		Whole at once	
	NSIT	SIT	NSIT	SIT
1	**38.10**	**38.10**	38.76	38.76
2	84.37	**84.00**	86.69	86.20
3	66.31	**64.46**	68.96	66.08
4	101.25	**96.89**	103.53	99.69
5	24.45	24.17	23.60	**22.90**
6	84.36	**83.37**	86.78	86.84
7	33.09	**32.47**	33.71	33.30
8	86.57	**83.26**	87.84	85.40
9	**57.57**	**57.57**	58.93	58.93
10	66.13	**63.74**	67.92	66.92

Table 6.5: *Distance [m] of the planned trajectory for the simulated model with the receding horizon and whole at once approaches, both initialized with and without smoothened initial trajectory (SIT and NSIT).*

Mission	Receding horizon		Whole at once	
	NSIT	SIT	NSIT	SIT
1	**47.78**	**47.78**	**47.78**	**47.78**
2	111.70	**111.14**	112.45	111.63
3	86.63	**83.98**	88.53	85.39
4	133.48	**129.12**	132.13	130.00
5	27.63	**26.59**	27.68	27.31
6	112.58	**110.47**	112.31	110.63
7	41.06	**40.56**	41.86	41.31
8	112.84	**110.11**	113.73	110.91
9	**74.61**	**74.61**	74.63	74.63
10	88.47	**83.59**	86.39	84.34

Table 6.6: *Distance [m] of the planned trajectory for the estimated model with the receding horizon and whole at once approaches, both initialized with and without smoothened initial trajectory (SIT and NSIT).*

Mission	Receding horizon		Whole at once	
	NSIT	SIT	NSIT	SIT
1	**47.78**	**47.78**	**47.78**	**47.78**
2	111.50	**111.02**	111.80	111.18
3	86.35	**83.74**	87.54	84.05
4	134.61	**128.76**	132.51	129.42
5	27.86	27.76	27.23	**26.66**
6	111.67	**110.10**	113.97	112.61
7	40.93	**40.09**	41.28	40.90
8	113.24	**109.90**	114.23	110.20
9	**74.61**	**74.61**	**74.61**	**74.61**
10	86.53	**83.21**	87.72	85.38

Table 6.7: *Numerical integration of $|n_1^3| + |n_2^3| + |n_t^3|$ [M/s^2] as a measure of energy for the simulated model with the receding horizon and whole at once approaches, both initialized with and without smoothened initial trajectory (SIT and NSIT).*

Mission	Receding horizon		Whole at once	
	NSIT	SIT	NSIT	SIT
1	**360.26**	**360.26**	361.50	361.50
2	526.72	517.43	488.19	**452.90**
3	993.99	641.61	586.36	**586.02**
4	769.96	637.64	**549.59**	618.33
5	715.50	578.14	**301.49**	583.89
6	688.44	572.98	590.23	**488.59**
7	898.48	**445.57**	473.74	453.14
8	669.20	647.17	547.67	**502.94**
9	408.80	408.80	**405.87**	**405.87**
10	621.44	601.70	591.13	**532.02**

Table 6.8: *Numerical integration of $|n_1^3| + |n_2^3| + |n_t^3|$ [M/s^2] as a measure of energy for the estimated model with the receding horizon and whole at once approaches, both initialized with and without smoothened initial trajectory (SIT and NSIT).*

Mission	Receding horizon		Whole at once	
	NSIT	SIT	NSIT	SIT
1	1621.69	1621.69	**1621.48**	**1621.48**
2	3713.27	**3696.56**	3737.03	3708.83
3	2896.87	**2813.23**	3000.06	2852.66
4	4476.19	**4280.04**	4496.65	4325.19
5	1037.47	1024.15	997.54	**963.27**
6	3716.69	**3681.36**	3801.43	3839.88
7	1405.68	**1378.81**	1424.04	1418.50
8	3690.66	**3663.06**	3791.54	3694.00
9	2501.82	2501.82	**2500.32**	**2500.32**
10	2904.21	**2791.72**	2963.31	2934.73

In Table 6.7 and Table 6.8, a measure of energy consumption over the planned trajectory is presented. The cube of the rotational speed of the thrusters are used, since it is proportional to the energy consumption according to Bärlund (2019). Contrary to previous tables for the simulated model, the receding horizon approach initialized with a trajectory based on smoothened path does not outperform the other approaches and initializations, whereas for the estimated model, the receding horizon approach initialized with SIT still performs better.

The vessel effectively avoids the obstacles, see Figure 6.6, Figure 6.7 and Figure 6.8, for all trajectories. But to keep the vessel away from the obstacles, a penalty on the distance between obstacles and the vessel could be added to the objective function J_d in (6.17) (Bergman et al., 2020).

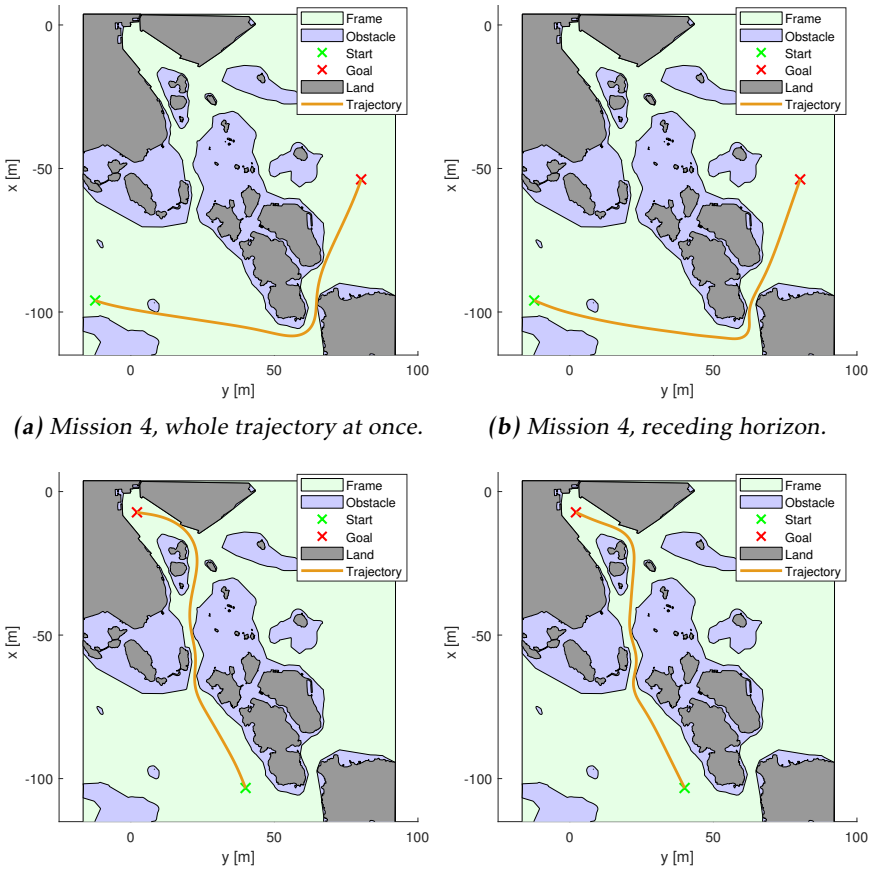
From Figure 6.6, it can be seen that the receding horizon optimization is straighter than when the trajectory is optimized all at once for the simulated model. This behavior is also seen in Figure 6.7 and Figure 6.8, for the estimated model.

In Figure 6.7 there are many relatively sharp turns in the trajectory and for Mission 8, the whole at once approach is even more curvy than the receding horizon approach. This is strange since the receding horizon approach only optimizes short segments at a time and is not aware of the whole trajectory. When comparing the estimated model initialized with a trajectory based on a smooth path (Figure 6.8), and without a smooth path (Figure 6.7), it can be seen that initializing with a smooth path also results in a trajectory that is closer to the obstacles. One reason for this is that the initial trajectory is smooth and closer to the boundary of the obstacles.

6.3.3 Conclusion

Since the estimated model the receding horizon approach takes unreasonably long time to calculate and since receding horizon approach is also slower for the simulated model, optimizing the whole trajectory at once is best suited to this problem. Initializing the optimization problem with a trajectory based on a smoothened path reduces the time, distance and energy of the planned trajectory in most of the cases for the estimated model and simulated model. Thus, initializing with a smooth trajectory increases the performance of the solver.

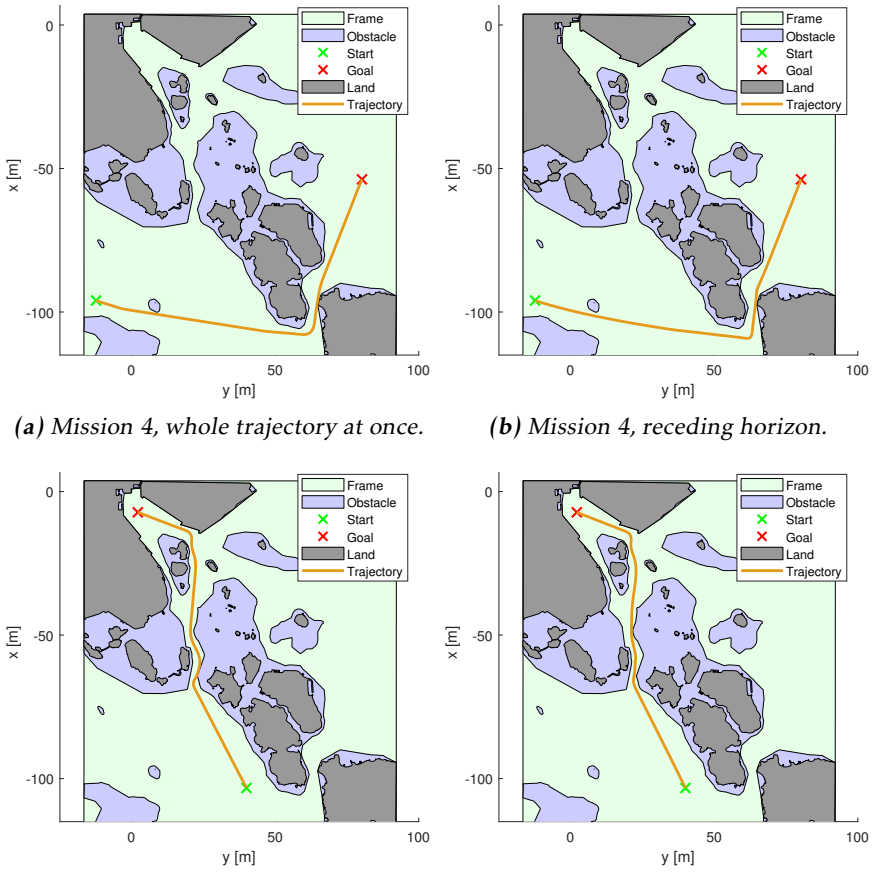
Why the receding horizon approach performs better than optimizing the whole trajectory at once is hard to say. Especially since the receding horizon approach should result in a suboptimal trajectory, while the other method should result in the optimal solution.



(a) Mission 4, whole trajectory at once. *(b) Mission 4, receding horizon.*

(c) Mission 8, whole trajectory at once. *(d) Mission 8, receding horizon.*

Figure 6.6: *Optimized trajectory for Mission 4 and 8 with the different optimization approaches using the simulated model and a non-smooth initial trajectory.*



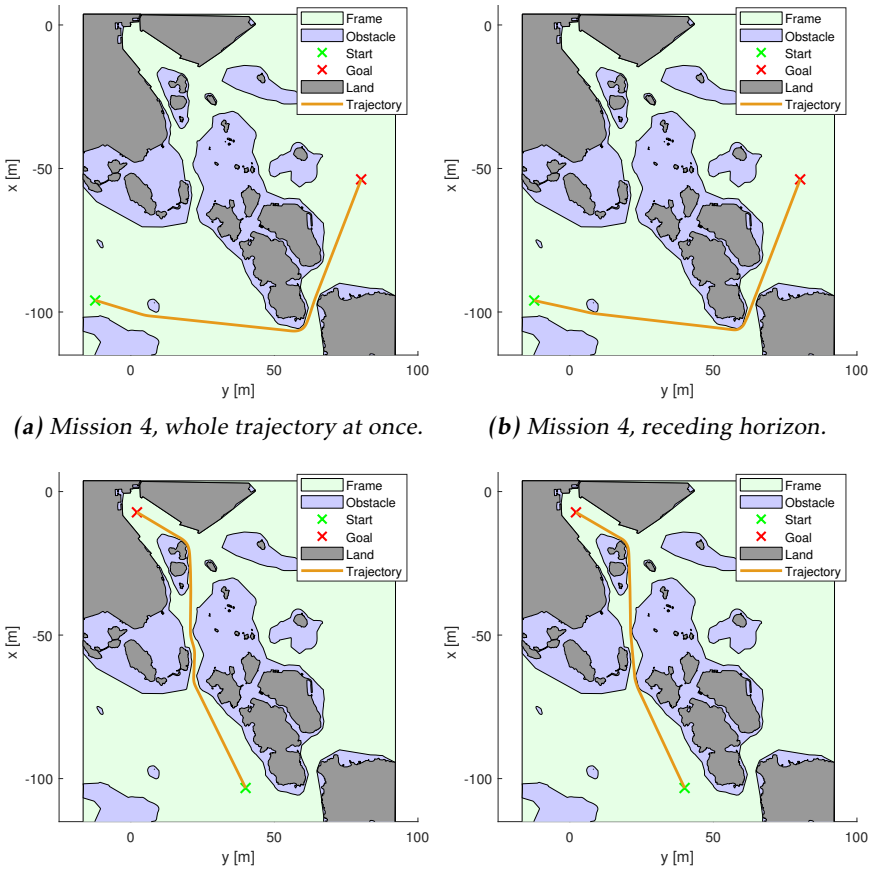
(a) *Mission 4, whole trajectory at once.*

(b) *Mission 4, receding horizon.*

(c) *Mission 8, whole trajectory at once.*

(d) *Mission 8, receding horizon.*

Figure 6.7: *Optimized trajectory for Mission 4 and 8 with the different optimization approaches using the estimated model and a non-smooth initial trajectory.*



(a) *Mission 4, whole trajectory at once.*

(b) *Mission 4, receding horizon.*

(c) *Mission 8, whole trajectory at once.*

(d) *Mission 8, receding horizon.*

Figure 6.8: *Optimized trajectory for Mission 4 and 8 with the different optimization approaches using the estimated model and a smooth initial trajectory.*

7

Conclusion and Future Work

In this thesis, a grey-box model for a ship has been presented, including the dynamics of the rudder and wind disturbances. The parameters in the grey-box model have been estimated using an optimization based system identification method, both on simulated data and experimentally. For the experimental data, the dynamics of the rudder and wind was neglected to help convergence of the NLP. Furthermore, a trajectory planning algorithm was proposed to obtain a dynamically feasible trajectory between a start and goal position given a map of obstacles and a dynamic model of a ship. Firstly, a roadmap is generated utilizing a Voronoi diagram. Then, a geometric path is found by employing the A*-algorithm to the roadmap. An initial guess for the trajectory optimization is obtained by mapping a straight line trajectory to the geometric path. From the optimization an optimal feasible trajectory is found from the start position to the goal, given the grey-box model and the estimated parameters, such that the trade-off between time duration and energy consumption is well-balanced.

7.1 Future Work

In this thesis, an system identification method has been used to find a dynamic model for the small-scale ship. Thereafter, trajectory planning has been developed, utilizing the dynamic model, to find a feasible path given a start and goal position and a set of obstacles. An intuitive future development would be to implement the given trajectory planning on the model ship along with a controller to evaluate its performance in real life applications.

From a system identification perspective it would be interesting to fit a sensor to measure the exact angle of the azimuth thruster, to get more accurate estimates of

the parameters and to validate the model in a more correct manner. Furthermore, an interesting approach would be to estimate the model parameters at different sampling frequencies to see if even more of the dynamics could be included in the model, such as rudder dynamics, wind dynamics and shallow water dynamics.

For the trajectory planning it would be interesting to see if the performance could be increased with a better initial guess of the trajectory. With a better initial guess of the trajectory, the NLP is less prone to get trapped at local minima and thus the planning time will be reduced. Moreover, it would be interesting to investigate the performance of the trajectory planning in departure and docking situations. Investigating this would make the trajectory planning algorithm more useful in real-life applications, where docking and undocking is unavoidable.

Appendix

A

Experiments

In this chapter, the experiments performed in the thesis are described, as well as a selection of the results.

A.1 System Identification Experiment

In this section the experiments performed to estimate the parameters of the ship model are described.

A.1.1 Goal

The goal of this experiment is to estimate the parameters ϑ in (2.24), with μ_{az} and μ_t fixed according to the result in Appendix A.2.

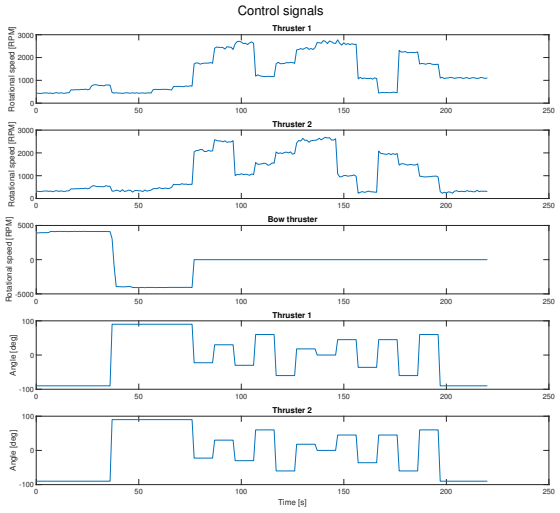
A.1.2 Procedure

The experiment is conducted by applying various input signals to the ship and measure its position and orientation. To estimate all parameters, manuevers at various speeds in all 3DOF, surge, sway and yaw-rate, are required. The procedure is therefore to operate the ship in a way that such data is obtained.

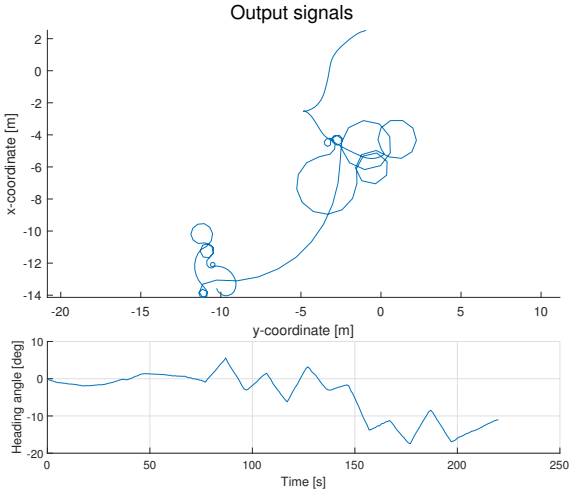
The experiment is preformed multiple times with various input signals.

A.1.3 Result

The collected data used as estimation and validation data can be seen in Figure A.1 and A.2, respectively.

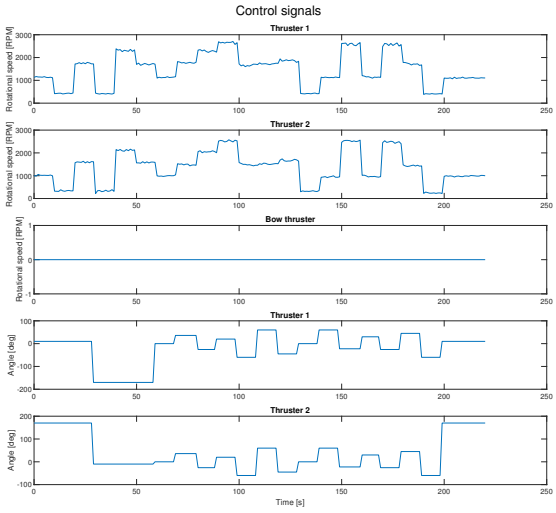


(a)

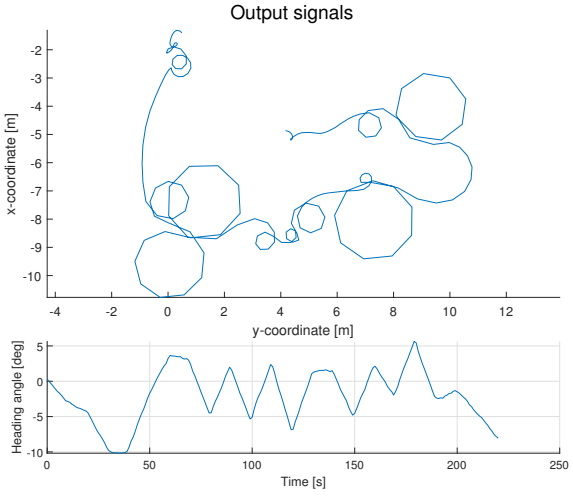


(b)

Figure A.1: *Experimental data used in estimation of the parameters. In (a) the control signals are presented and in (b) the outputs can be seen.*



(a)



(b)

Figure A.2: *Experimental data used in estimation of the parameters. In (a) the control signals are presented and in (b) the outputs can be seen.*

The resulting parameters can be seen in Table 4.3 and more details about the result can be seen in Section 4.5.2.

A.2 Thruster Identification Experiment

This section describes the experiment to estimate the thruster parameters on the small-scale ship. This type of experiment is usually called a Bollard Pull test.

A.2.1 Goal

The goal is to estimate the model parameters μ_{az} and μ_t for the thrusters in (2.13) and (2.15). The total trust produced by each thruster in stationarity can be expressed with

$$\tau_i = \mu_i n_i^2. \tag{A.1}$$

A.2.2 Procedure

The test is conducted by attaching a dynamometer between the ship and a fixed object, thereafter the thrust for each thruster is measured at various rotor speeds. During experiments it was found out that the dynamometer could not read forces below 1 N, therefore it was not possible to measure the force at low rotor speeds. Furthermore, the bow thruster did not produce a force higher than 1 N, making its effect impossible to measure. Hence, an experiment was designed where one azimuth thruster with $\alpha_i = \frac{\pi}{2}$ was used in combination with the bow thruster to generate a force in y_b-axis. When the ship had a pure sway motion the rotor speeds of the thrusters were noted. Thereafter, knowing the force from the azimuth thruster, the force produced by the bow thruster could be calculated. Lastly a least square curve fitting method was used to estimate μ_{az} and μ_t in (A.1) from the data.

A.2.3 Result

The results from the Bollard pull tests are described below.

Azimuth thrusters

For the azimuth thrusters the force at various rotor speeds can be observed in Figure A.3. The model (A.1) is fitted to the measurements, and the parameter μ_{az} is found to be $1.435 \cdot 10^{-6}$.

Bow thruster

For the bow thruster a pure sway motion was found at the rotor speeds shown in Table A.1.

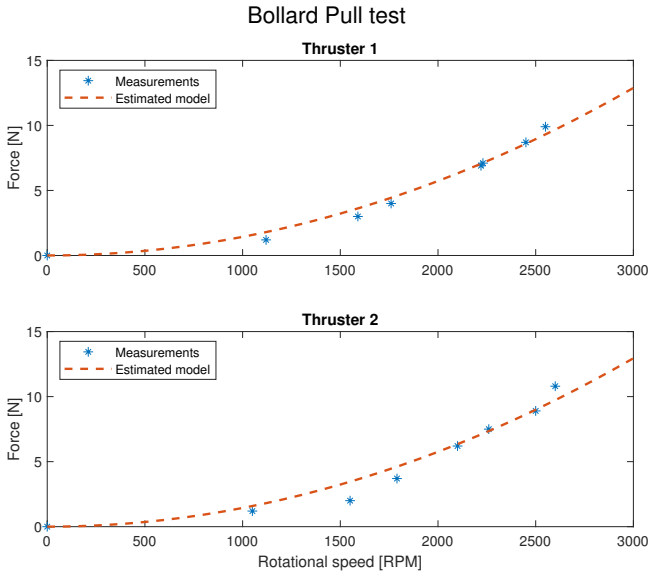


Figure A.3: *Results from the Bollard pull test of the azimuth thrusters.*

Table A.1: *Measured rotor speeds to achieve a pure sway motion.*

Thruster 2	**Bow thruster**
460 RPM	2390 RPM
550 RPM	2840 RPM
735 RPM	4060 RPM

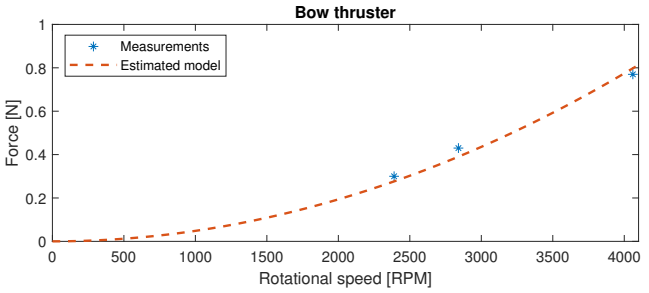


Figure A.4: *Results from the Bollard pull test of the bow thrusters.*

The thrust produced by the bow thruster was thereafter calculated using the known μ_{az}, and the result can be seen in Figure A.4. The model (A.1) is fitted to the measurements, and the parameter μ_t is found to be $4.840 \cdot 10^{-8}$.

A.3 PWM to Angle Mapping

Due to the lack of measurements for the angle of the azimuth thruster, in this section experiments are performed to estimate the angle on the azimuth thrusters given a pulse width modulation (PWM) signal from the controller.

A.3.1 Goal

The goal of this experiment is to estimate the angle on the azimuth thrusters given a PWM signal.

A.3.2 Procedure

The experiment is conducted by applying PWM signals with various pulse widths to the servos actuating the angle of the azimuth thrusters. Thereafter, a least square curve fitting method is used to find the relationship between the pulse width and the angle of the azimuth thruster.

A.3.3 Result

During the experiment it was found that the angle of the azimuth thrusters varied approximately 20 degrees with the same pulse width of the PWM signal. This was probably due to slack in the drive belts or due to resistance in the servos. Therefore, the mapping was done from low to high pulse width and vice versa, obtaining the result shown in Figure A.5. A line has also been fitted to the measured data.

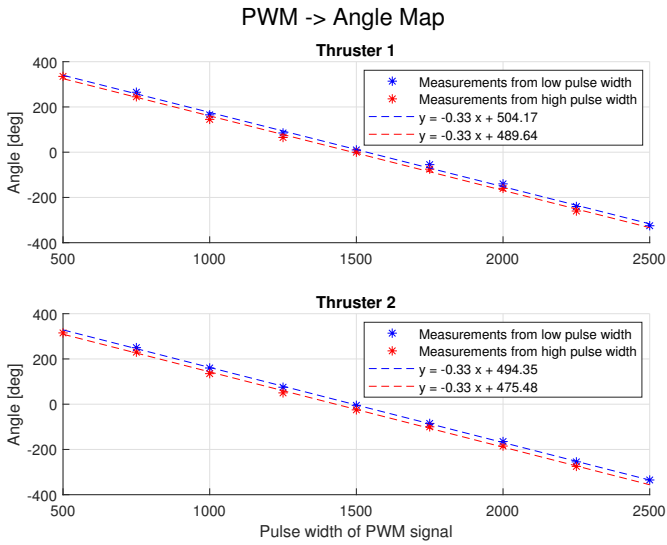


Figure A.5: *Results from the PWM to Angle mapping of the azimuth thrusters.*

B

Proofs

In this appendix, various proofs are presented.

B.1 Shortest Distance Between Figures (Obstacles)

Proof of shortest distance between figures: Based on Figure 5.1, let A and B be two generator points on a figure g_i separated by the distance d. Let C be the generator point on another figure g_j that is closest to AB and denote the projection of C onto AB with D. Also let E be the Voronoi vertex generated by A, B and C and let F be its projection onto AB. With

$$a = |AD|, \quad b = |BD|, \quad c = |DF|, \quad d = |AB|$$
$$x = |CD|, \quad y = |EF|, \quad z = |AE| = |BE| = |CE|$$

then

$$z^2 = c^2 + (x - y)^2 \tag{B.1a}$$
$$z^2 = y^2 + (a + c)^2 \tag{B.1b}$$
$$z^2 = y^2 + (b - c)^2 \tag{B.1c}$$

Equation (B.1a) inserted in (B.1b) gives

$$c^2 + (x - y)^2 - y^2 - (a + c)^2 = 0 \quad \overset{a>0}{\Rightarrow} \quad c = \frac{x^2 - 2xy - a^2}{2a} \tag{B.2}$$

Then, subtracting (B.1c) from (B.1b) gives

$$\begin{aligned} (a + c)^2 = (b - c)^2 \quad &\Rightarrow \quad 2(a + b)c = b^2 - a^2 \\ &\overset{a+b>0}{\Rightarrow} \quad c = \frac{b - a}{2} \end{aligned} \tag{B.3}$$

Now, inserting (B.3) in (B.2) gives

$$\frac{b-a}{2} = \frac{x^2 - 2xy - a^2}{2a} \quad \Rightarrow \quad y = \frac{x^2 - ab}{2x} \tag{B.4}$$

For the Voronoi vertex E to lie between the figures g_i and g_j, $0 < y < x$. With this condition, the closest distance x to g_j given the distance d between generator points on g_i can be calculated using $y > 0$ and (B.4) as

$$\frac{x^2 - ab}{2x} > 0 \quad \Rightarrow \quad x > \sqrt{ab} = \sqrt{a(d-a)}. \tag{B.5}$$

□

B.2 Worst Generator Point Placement

Proof of worst generator point placement: The worst case, i.e., when x in (B.5) has to be largest can be calculated by setting the derivative $\frac{d}{da}(a(d-a)) = 0$,

$$\begin{aligned} \frac{d}{da}(a(d-a)) = d - 2a = 0 \quad &\Rightarrow \quad a = d/2 \\ &\Rightarrow \quad a = b = d/2 \end{aligned} \tag{B.6}$$

and since

$$\frac{d^2}{da^2}(a(d-a)) = -2 < 0$$

this is indeed a maximum. Thus, x has to be the largest where $a = b$. □

Bibliography

J. A.E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi: 10.1007/s12532-018-0139-4.

K. Bergman, O. Ljungqvist, J. Linder, and D. 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. IEEE, 2020.

P. Bhattacharya and M. L. Gavrilova. Geometric algorithms for clearance based optimal path computation. In *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, 2007. doi: 10.1145/1341012.1341064.

G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik. Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments. *IEEE Access*, 8:199953–199969, 2020. doi: 10.1109/ACCESS.2020.3035256.

H.G. Bock and K.J. Plitt. A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems. *IFAC Proceedings Volumes*, 17(2), 1984. doi: 10.1016/S1474-6670(17)61205-9.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 9780511804441. doi: 10.1017/CBO9780511804441.

A. Bärlund. Nonlinear MPC for Motion Control and Thruser Allocation of Ships. Master's thesis, Linköping University, 2019.

B. H. Eriksen and M. Breivik. MPC-Based mid-level collision avoidance for ASVs using nonlinear programming. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, 2017. doi: 10.1109/CCTA.2017.8062554.

European Maritime Safety Agency. Annual Overview of Marine Casualties and Incidents 2020. 2020.

T. I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Incorporated, first edition, 2011. ISBN 978-1-119-99149-6.

Masahiko Furuichi and Ryuichi Shibasaki. Cascade strategy of container terminals to maximize their quantitative and qualitative capacity. In *Proceedings of IAME 2015 Conference, Kuala Lumpur, Malaysia*, 2015.

F. Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur, Lund, 2018. ISBN 978-91-44-12724-8.

P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.

U. Jönsson, C. Trygger, and P. Ögren. *Optimal Control - Lecture notes*. Royal Institute of Technology, Stockholm, 2010.

D. Kang, V. Nagarajan, K. Hasegawa, and M. Sano. Mathematical model of single-propeller twin-rudder ship. *Journal of Marine Science and Technology*, 13, 2008. doi: 10.1007/s00773-008-0027-0.

S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006. ISBN 978-0-521-86205-9.

E.M. Lewandowski. *The Dynamics of Marine Craft: Maneuvering and Seakeeping*. Advanced series on ocean engineering. World Scientific, 2004. ISBN 9789810247560.

L. Ljung and T Glad. *Modellbygge och simulering*. Studentlitteratur, Lund, 2004. ISBN 978-91-44-02443-1.

F. Ljungberg. *Estimation of Nonlinear Greybox Models for Marine Applications*. Licentiate thesis no. 1880, Department of Electrical Engineering, Linköping University, 2020.

R. Mahony, T. Hamel, and J.-M. Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, pages 1203–1218, 2008. doi: 10.1109/TAC.2008.923738.

P. Mucha. *On Simulation-based Ship Maneuvering Prediction in Deep and Shallow Water*. PhD thesis, University of Duisburg-Essen, 2017.

B.K. Patle, G. Babu L, A. Pandey, D.R.K. Parhi, and A. Jagadeesh. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 15(4):582–606, 2019. doi: 10.1016/j.dt.2019.04.011.

F. Peralta, M. Arzamendia, D. Gregor, D. G. Reina, and S. Toral. A Comparison of Local Path Planning Techniques of Autonomous Surface Vehicles for Monitoring Applications: The Ypacarai Lake Case-study. *Sensors*, 20(5), 2020. doi: 10.3390/s20051488.

SNAME. *Nomenclature for Treating the Motion of a Submerged Body Through a*

Fluid. Technical and research bulletin. Society of Naval Architects and Marine Engineers, 1952.

M.W. Spong and S. Hutchinson. *Robot Modeling and Control*. Wiley, 2005. ISBN 9780471649908.

K. Sugihara. Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams. *CVGIP: Graphical Models and Image Processing*, 55(6):522–531, 1993. doi: 10.1006/cgip.1993.1039.

N. W. Teeuwen. On the determination of hydrodynamic coefficients for real time ship manoeuvring simulation. Master's thesis, Delft University of Technology, 2018.

UNCTAD. *Review of Maritime Transport 2020*. United Nations, 2020. ISBN 978-92-1-112993-9.

A. Vagale, R. T. Bye, R. Oucheikh, O. L. Osen, and T. I. Fossen. Path Planning and Collision Avoidance for Autonomous Surface Vehicles I: A Review. *Journal of Marine Science and Technology*, 2020. doi: 10.1007/s00773-020-00787-6.

A. Wächter and L. Biegler. *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*. PhD thesis, Carnegie Mellon University, 2005.

M. Wingrove. Industry sails course towards autonomous shipping. 2020. URL `https://www.rivieramm.com/news-content-hub/news-content-hub/industry-sails-course-towards-autonomous-shipping-60359`. Online; accessed 10 June 2021.

Bin Xu, Daniel J Stilwell, and Andrew J Kurdila. A receding horizon controller for motion planning in the presence of moving obstacles. In *2010 IEEE International Conference on Robotics and Automation*, pages 974–980. IEEE, 2010.