NTNU
Norwegian University of
Science and Technology

Automated berthing (parking) of autonomous ships

Ketil Grav Skjåstad

Master of Science in Cybernetics and Robotics

Submission date:	June 2018
Supervisor:	Kristin Ytterstad Pettersen, ITK
Co-supervisor:	Matko Barisic, ABB

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Preface

This thesis is the result of a one-year collaboration between NTNU and ABB AS. ABB has provided me with a detailed MATLAB Simulink hydrodynamic ship simulation model. This model includes thrust allocation, actuator saturation, and realistic propulsion blocks. I have received MATLAB code for Matko Barisic VPM method from his Ph.D., which I have had to completely rewrite to be used in Simulink.

This thesis is based on my project thesis which started in August 2017 and ended in December 2017. The relevant discussions and results presented in the project thesis are included here. These include the path planner algorithm, its associated theory, and it's results, and most of the literature review.

I would like to thank my supervisors Kristin Y. Pettersen and especially Matko Barisic for his help and support during all stages of this thesis. I would also like to thank Philipp Nguyen at ABB for providing the Simulink ship model walking me through how to use it.

Abstract

Shipping operations can increase their efficiency by automating standard operations. This thesis explores the concept of automated navigation in a static harbor environment and automating the berthing procedure for a commercial ship. The ship is actuated with two stern azimuth thrusters and a bow tunnel thruster, giving it full maneuverability.

A literature study is done on berthing procedures, collision avoidance systems, and path following control. Several methods of collision avoidance are evaluated. An A-Star (A*) algorithm for path planning has been implemented and extended upon. A path following kinematic controller has been implemented in order to steer the ship along the planned path. The Virtual Potential Method (VPM) has been implemented collision avoidance with stationary but unforeseen obstacles, not accounted for by the path planner. Finally, a nonlinear PID Dynamic Positioning (DP) controller has been implemented to steer the ship the final distance to the berth.

Simulations have performed for a detailed hydrodynamic ship model by use of MATLAB Simulink. The results show that the A* method with extensions is a suitable path planning tool. The path following controller and the VPM for collision avoidance perform satisfactorily but are not robust. The DP controller does not perform satisfactorily as of now. Improvements for all the methods are suggested for future work.

Sammendrag

Maritim transport kan øke effektiviteten ved å automatisere vanlige operasjoner. Denne rapporten utforsker det å automatisere navigasjon i et statisk havnemiljø og det å legge skipet til kai. Et kortreisende transportskip er brukt som basis. Skipet drives av to azimuth-thrustere akter og en tunnel-thruster i baugen. Dette gir skipet full bevegelighet.

En litteraturstudie er gjort på automatisering av det å legge til kai, kollisjonsunngåelse og banefølging. Flere metoder for kollisjonsunngåelse blir vurdert. En A* algoritme for baneplanlegging er implementert og utviklet videre. En banefølgende regulator er implementert for å sørge for at skipet blir stryrt langs den planlagte banen. VPM er brukt for kollisjonsunngåelse med stasjonære, men uforutsette hindringer som ikke er blitt tatt hensyn til av baneplanleggeren. Til slutt er en ulineær PID regulator brukt for DP. Denne brukes til å styre skipet den siste distansen til kaia.

Simuleringer har blitt gjennomført på en detaljert hydrodynamisk skipsmodell ved bruk av MATLAB Simulink. Resultatene viser at A* metoden for baneplanlegging fungerer godt. Regulatoren for banefølging og bruk av VPM for kollisjonsunngåelse viser tilfredsstillende resultater, men de er ikke robuste. DP regulatoren gir ikke tilfredsstillende resultater. Forbedringer til metodene er foreslått i videre arbeid.

Table of Contents

List of Tables

List of Figures

List of Acronyms

DW Dynamic Window

RPM Revolutions Per Minute

SF Serret-Frenet

DP Dynamic Positioning

LOS Line Of Sight

NED North-East-Down

LP Low Pass

TA Thrust Allocation

FOV Field Of View

MPC Model Predictive Control

POA Projected Obstacle Area

COLREGS International Regulations for Preventing Collisions at Sea 1972

A* A-Star

RRT Rapidly-exploring Random Trees

UUV Unmanned Underwater Vehicle

USV Unmanned Surface Vehicle

LIDAR Light Detection and Ranging

SLAM Simultaneous Localization and Mapping

APF Artificial Potential Field

VPM Virtual Potential Method

DOF Degrees Of Freedom

Chapter 1

Introduction

1.1 Motivation

For our entire history, humans have always sought ways to make life easier for ourselves, to be more effective. From making basic tools to organized agriculture to steam power and so on. We always strive to do more with less. Automation has been the next step in efficiency. Starting with the industrial revolution, manual workers have been replaced by machines and put in positions of supervision and management.

Today's next step is to automate transport and travel. Huge progress has been made in this area already. Autopilots of all kinds dominate air-travel and shipping. There are self-driving cars fully capable of driving in public, and many countries are allowing test trials for them [BBC (2014), E.U.CORDIS Research Program CitynetMobil. (2013)].

One major problem with autonomous transport is the lack of a legal framework for operation. In civilian use, it is important that the unmanned systems are well documented to be equally or more safe than the manned equivalent. There are issues with liability in case of accidents. It is important to have good legal frameworks in place regarding safety and regulations before autonomous civilian operations can become commonplace.

The aim of this thesis is to create an autonomous berthing system for a commercial, short sea shipping vessel. This system will be able to autonomously bring the vessel from the harbor all the way to the quay. Such a system will benefit from the saving of cutting out the human operator or allowing one remote operator to oversee the operations of several ships. Similarly to automatic landing systems of modern airplanes, it will ease the process of berthing in bad weather.

1.2 Problem definition

The autonomous berthing system must be able to guide the vessel from entry to the harbor to its preallocated berthing spot. In order to achieve this autonomously, the system will need the capability to plan a path from its current position to its goal, while avoiding collisions with the static harbor environment and considering the constraints of the vessel dynamics. In addition, it must be able to avoid dynamic obstacles such as other ships in the harbor and other objects not accounted for by the path planner system, following the International Regulations for Preventing Collisions at Sea 1972 (COLREGS). The system should be simulated on a detailed vessel model to prove its functionality.

1.2.1 Assumptions

It is assumed that the system is provided with an accurate and detailed map of the harbor environment it is operating within. This map should include most static obstacles the vessel may encounter such as docks, quays, berthed ships, shallows, etc. The vessel is also assumed to be equipped with a sensor package capable of sensing dynamic obstacles such as traffic in the harbor, as well as any undocumented static obstacles. The positions and velocities of these dynamic obstacles are assumed to be available. This means that any obstacle present in the environment within the system's sensor range is known to the system. The vessel assumes full knowledge of its state variables such as accelerations, velocities, position, and attitudes.

1.2.2 Autonomy

There are levels of automation published by SAE International. While these are meant for the automotive industry they are applicable to the maritime as well. Table 1.1 and 1.2 gives the SAE (J3016) autonomy levels [SAE International (2016)]. The ultimate goal of the autonomous berthing system would be to operate on SAE level 5, where there is no need for any human input in any case. Present legal framework, and likely the operating company, would require a human operator in the loop. This thesis will have the goal to create a system in the SAE 3 category. In this category, the system will be able to handle any situation, like emergency maneuvers, without the operator paying attention. The operator is expected to be present and able to be called upon by the system within a limited time frame, to for example approve of a generated path.

1.3 System overview

The problem presented in section 1.2 is in this thesis broken down into four different parts. These are *path planning*, *path following*, *dynamic obstacle avoidance* and *berthing*. Each of these are handled separately. The ship model is presented first in order to explain the reasoning behind this breakdown.

1.3.1 Ship model

The simulation model of the ship is provided by ABB AS. The ship has a length of $294\,\mathrm{m}$, width of $37.9\,\mathrm{m}$ and mass of $44\,000\,000\,\mathrm{kg}$. The ship is equipped with two stern azimuth thrusters and a bow tunnel thruster as shown in figure 1.1. The azimuth thrusters can rotate and provide thrust in any direction and independently. The rotation is not instantaneous and incurs a significant time delay in actuation. The ship model comes with a complete Thrust

Table 1.1: SAE Autonomy Levels (1/2)

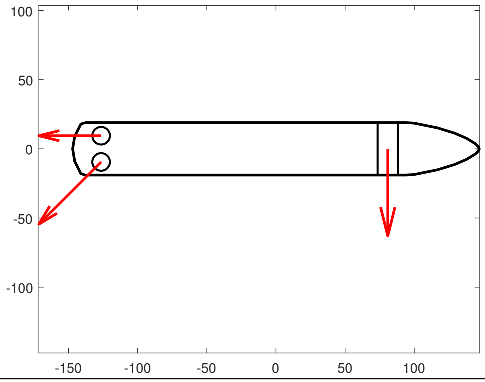
SAE Level/ Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	Monitoring of Driving Environment	Fallback Performance of Dynamic Driving Task
Human driver monitors the driving environment				
0/ No Automation	The full-time performance by the human driver of all aspects of the dynamic driving task, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver
1/ Driver Assistance	The driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task	Human driver and system	Human driver	Human driver
2/ Partial Automation	The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task	System	Human driver	Human driver

Table 1.2: SAE Autonomy Levels (2/2)

SAE Level/ Name	**Narrative Definition**	**Execution of Steering and Acceleration/ Deceleration**	**Monitoring of Driving Environment**	**Fallback Performance of Dynamic Driving Task**
Automated driving system monitors the driving environment				
3/ Conditional Automation	The driving mode-specific performance by an Automated Driving System of all aspects of the dynamic driving task with the expectation that the human driver will respond appropriately to a request to intervene	System	System	Human driver
4/ High Automation	The driving mode-specific performance by an Automated Driving System of all aspects of the dynamic driving task, even if a human driver does not respond appropriately to a request to intervene	System	System	System
5/ Full Automation	The full-time performance by an Automated Driving System of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver	System	System	System

Allocation (TA) system. This system includes actuator saturation, time delay in rotation of thrusters, and a realistic propulsion model. The TA system is discussed further in section 3.2.3.

Figure 1.1 Thruster layout on ship model



The thruster layout of the ship allows for full maneuverability. However, a ship of this size has considerably slow dynamics and huge cost associated with braking and acceleration, and movement in sway direction. Therefore it is preferable for the ship to spend the majority of its time at cruise speed in surge direction.

1.3.2 Path planning

The path planner uses the map of the environment in order to plan the shortest path from the current position of the ship and to the berth, while avoiding obstacles. The planner takes the dynamics of the ship discussed above into account. It assumes the ship is traveling at cruise speed when following the path, and therefore the curvature of path cannot exceed the turning radius of the ship at cruise speed. The planner also plans the path with a safety margin with respect to proximity to obstacles and land. The path planner is discussed in chapter 4.

1.3.3 Path following

The path following controller steers the ship along the path provided by the path planner. It controls the yawrate of the ship. The yawrate command is based based on deviation from the path, curvature of the path and current velocity. A speed controller ensures that the ship is always travelling at cruise speed. The path following controller is discussed in chapter 5.

1.3.4 Dynamic collision avoidance

The dynamic collision avoidance method ensures that the ship can react to the detection of obstacles in its path. These may include other ship traffic, or obstacles not included in the environmental map. When an obstacle is sensed it is fed to the dynamic collision avoidance system which returns a yawrate command. This command is based on the proximity of the obstacle and the direction towards the path. Dynamic collision avoidance is discussed in chapter 6.

1.3.5 Berthing

When the ship has successfully followed the planned path it should be located close to its berth. The berthing phase consists of steering the ship in the sway direction at a constant heading towards the berth. This system utilizes the full maneuverability of the ship model. It is important to keep the heading constant and aligned with the quay. A small heading deviation may cause the stern or bow to crash into the quay. The berthing controller is discussed in chapter 7

1.4 Contribution

The contribution of this thesis is the investigation of the four different methods presented in section 1.3 and their ability to solve the problem presented in section 1.2.

1.5 Outline

The report is organized as follows. Chapter 2 is a literature study of the requirements of the completed berthing system is presented. Chapter 3 is a short introduction to the theory of marine craft motion and control. It also introduces theory on collision avoidance methods. Chapter 4 develops the path planning method, detailing the implementation and modifications made to it. In chapter 5 the path following controller is presented. Chapter 6 details the dynamic collision avoidance method and how it uses the VPM method to generate its control signals. The DP controller used for the berthing procedure is presented in chapter 7. Simulation results are presented and discussed in chapter 8. The tesis concludes with chapter 9, where future work is discussed.

Chapter 2

Literature review

2.1 Berthing

A plethora of methods for berthing control exist. There are two phases to berthing, the first phase is called the *ballistic phase* and the second is called the *final phase* [Djouani and Hamam (1995)]. The ballistic phase is where only the propeller and rudder are used until the ship is sufficiently close to the berthing spot. In the final phase, the side thrusters or azimuth thrusters are used as well to move the ship laterally towards the dock.

Underactuated berthing

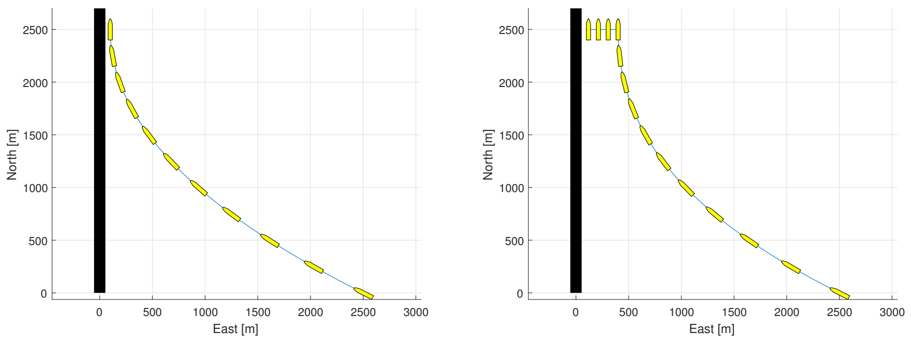
In the case where the ship is underactuated, meaning it only has a stern propeller, only the surge and heading of the ship is controllable. In this case, the ballistic phase is the whole approach. The general approach in this situation is shown on the left in figure 2.1. For underactuated ships without side thrusters, this motion is very complicated and most motion planning methods require very precise mathematical models of the ship and the environmental disturbances. The environmental effects of wind, waves, currents, and disturbances from shallow waters and bank effects have large impacts on maneuverability at the low speeds of a berthing maneuver [Bu et al. (2007)].

Okazaki and Ohtsu (2008) presents a solution and an actual sea test of a minimum time berthing controller. The solution is based on a sophisticated non-linear mathematical model and is transformed into waypoints to be used with traditional waypoint navigation. Yao et al. (1997) develops a multivariable neural controller for berthing which has no need for a mathematical model of the ship. The berthing path isn't restrictive and can be generalized to fully actuated berthing as well. Bu et al. (2007) presents a sliding mode control trajectory planner and a feedback control is presented to guide the ship along a simple path.

Fully actuated berthing

In the case where the ship is equipped with lateral thrusters or azimuth thrusters, the problem is simplified. Sway motion is now controllable. The situation can be seen on the right in figure 2.1. In this case, when the ballistic phase is solved and completed, a standard DP system with a slowly varying reference can be used to complete the final phase of the berthing. In [Djouani and Hamam (1994)] a neural network controller is suggested to solve the final phase. In [Yao et al. (1997)] a multivariable neural controller is used for an underactuated ship in berthing, but it is proposed to be expanded to fully actuated ships. A multitude of DP controllers are discussed in [Fossen (2011)] ranging from simple PID control to more complicated optimal controllers.

Figure 2.1 Berthing maneuvers



Relevance to this thesis

The cases of underactuated berthing provide context and understanding to the complexities of the berthing problem. In the context of this thesis, it is unnecessary to implement an underactuated berthing controller, as the system will have access to azimuth thrusters and will be fully actuated. ABB have already developed robust DP controllers. When making the complete autonomous berthing system it is preferable to use the DP controller from ABB. As a proof of concept implementing a simple DP controller from [Fossen (2011)] is attempted in this thesis.

2.2 Collision avoidance

Djouani and Hamam (1994) presents optimal path planning with collision avoidance of a berthing maneuver as a non-linear mathematical optimization problem. The problem uses a highly detailed mathematical model of the vessel with state and actuator constraints, and non-linearities. The path planner generates a feasible path for the detailed model of the vessel off-line.

Loe (2007) presents a thorough review of collision avoidance methods. A handful of global and local methods are simulated and compared in their ability to provide effective collision avoidance for an Unmanned Surface Vehicle (USV). The combination of the Dynamic Window (DW) method and either the A* or Rapidly-exploring Random Trees (RRT) method is concluded to provide the best system. Constrained nonlinear optimization as discussed by Djouani and Hamam (1994) above is concluded to be too troublesome for implementation. The main issues for this being difficulty in guaranteeing global solutions when the problem is non-convex, and inability to find a feasible solution.

Implementations of CA

Loe (2008) follows up on his previous work and uses the RRT and DW methods to create a collision avoidance system for simulation and implementation on a full scale, real USV. Ueland et al. (2017) implements a system of marine autonomous exploration for USVs. The system uses information of its surroundings provided by Light Detection and Ranging (LIDAR) for Simultaneous Localization and Mapping (SLAM). Using this map it generates a path to the unexplored frontier using the A* method.

Another system of collision avoidance is presented by Larson et al. (2006) also using the A* method for path planning. This method is modified to plan its route around a Projected Obstacle Area (POA). These are areas where moving obstacles are predicted to be in the future based on the relative speeds of the vessel and the obstacle. A reactive obstacle avoidance scheme is added as the path planner cannot guarantee collision avoidance even with POA. Kørte (2011) considers guidance and control of Unmanned Underwater Vehicle (UUV) and focuses on local methods when considering collision avoidance.

Barisic (2012) presents a method for coordinated control of a formation of UUVs using an Artificial Potential Field (APF) method called the VPM. The VPM method can be adapted to be used for a singular USV.

Relevance to this thesis

The review presented by Loe (2007) provides a thorough understanding of current methods of collision avoidance. Proof of the A* method's performance is seen in the in the two different implementations. This section then provides a good basis for the choice of methods needed to solve the problem of dynamic collision avoidance in this thesis. The POA method by Larson et al. (2006) is useful for future work.

2.3 Path following

When a path has been planned to guide the ship free of collisions, a controller must be implemented to ensure that the path is followed.

Guidance

Chapter 10 in Fossen (2011) explores methods for design of guidance systems for marine craft. A distinction is made between time-variant *trajectory tracking* and time-invariant *path following*. The guidance systems generate references to the motion controller to regulate the vessel toward its trajectory or path. The simplest form of path following is LOS based methods when following straight lines between waypoints. These seek to point the ships heading towards a point on the line a certain distance ahead. When the ship is sufficiently close to a waypoint the next waypoint is selected and the next line is followed. This circle of acceptance is recommended to have a radius of $2L_{pp}$, where L_{pp} is the length of the ship. Therefore the waypoints must be spaced far enough apart for this method to be viable. For parameterized paths, a path following kinematic controller is considered. Ueland et al. (2017) uses a similar controller to this one, generating a set-point a point ahead on the parameterized path which is sent to the motion controller. The set-point's distance ahead on the path is dependent on how close the vessel is to an obstacle, which limits corner-cutting when far off the path.

Skjetne et al. (2005) formulates the maneuvering problem. It is divided into tasks, the first called the geometric task which to force the system towards the desired path. The second is called the dynamic task which is to satisfy a desired speed along the path. The geometric task is given higher priority in the solution of the maneuvering problem. This merges the path following and path tracking as discussed above. It assumes a smooth parameterized path with bounded first and second partial derivatives.

Relevance to this thesis

The path following solution must be designed to suit the path provided by the path planner/collision avoidance system. If the path consists of waypoints, the straight line methods may be viable. However, if the spacing of the waypoints is too close these methods are is not viable. An interpolation method is used in Ueland et al. (2017) to interpolate and generate a smooth, parameterized path between the waypoints, and thus makes the path following kinematic controller viable. In conclusion, the path following kinematic controller was decided to be implemented as the planned path will be smooth and parameterized.

Chapter 3

Theory

This chapter will present the equations of motion for marine craft as formulated in Fossen (2011). Then it will introduce and briefly discuss the collision avoidance methods reviewed in Loe (2007). The method to be implemented in chapter 4 is based on this discussion. Local methods and the path following controller are introduced and discussed for their use in future work.

3.1 Equations of motion for marine craft

The equations of motion for the marine craft are used in the MATLAB Simulink model provided by ABB to create a realistic response to control inputs. The model uses Fossen's nonlinear 6 Degrees Of Freedom (DOF) vector equations. The vessel states are its general coordinates and attitude, $\boldsymbol{\eta} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}$, given in the inertial frame $\{i\}$, and its linear and angular velocities, $\boldsymbol{\nu} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}$, given in the body frame $\{b\}$.

The equations of motion expressed in $\{b\}$ are given by:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_\Theta(\boldsymbol{\eta})\boldsymbol{\nu} \tag{3.1}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{g}_0 = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}} \tag{3.2}$$

where:

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A \tag{3.3}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}) \tag{3.4}$$

$$\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D} + \boldsymbol{D}_n(\boldsymbol{\nu}) \tag{3.5}$$

- $\boldsymbol{M}_{RB}$ is the rigid body inertia matrix
- $\boldsymbol{M}_A$ is the hydrodynamic added mass matrix
- $\boldsymbol{C}_{RB}(\boldsymbol{\nu})$ is the rigid body Coriolis and centripetal matrix

- $\boldsymbol{C}_A(\boldsymbol{\nu})$ is the hydrodynamic Coriolis and centripetal matrix
- $\boldsymbol{D}$ is the linear damping matrix.
- $\boldsymbol{g}(\boldsymbol{\eta})$ and $\boldsymbol{g}_0$ are hydrostatic restoring and ballast forces and moments.
- $\boldsymbol{D}_n(\boldsymbol{\nu})$ is the nonlinear damping matrix
- $\boldsymbol{J}_\Theta(\boldsymbol{\eta})$ is the Euler angle rotation matrix
- τ is the control inputs vector $\tau = \begin{bmatrix} X & Y & Z & K & M & N \end{bmatrix}$
- $\boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}}$ are the environmental disturbances to to wind and waves

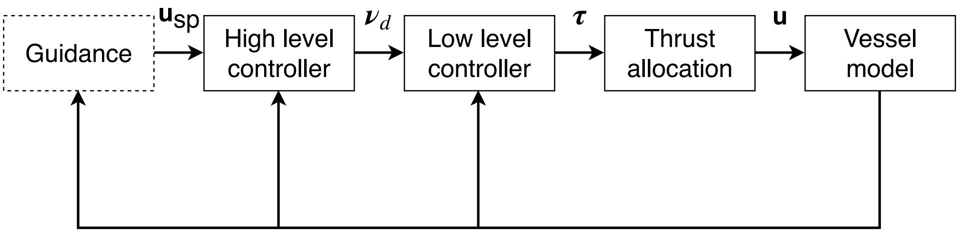
These equations of motion are the basis of the detailed Simulink model provided by ABB and explained in detail in Fossen (2011).

3.2 Low level control

The nature of the higher level controllers for path following and dynamic collision avoidance presented later in chapters 5 and 6 is to command desired speed and yaw/turn-rate. The low level controllers work to provide a desired force or moment in order to satisfy the desired speed or turn-rate given by the high level controllers. The desired forces and moments are then sent to the thrust allocation block to drive the actual thrusters and propellers.

The control flow of the system is summarized in figure 3.1. $\boldsymbol{u}_{sp}$ is the set-point given by the guidance system, $\boldsymbol{\nu}_d$ is the desired speed/turn rate, $\boldsymbol{\tau}$ is the desired forces and moments, and $\boldsymbol{u}$ are the commanded RPM and angles of the azimuth thrusters and bow tunnel thruster. The vessel states $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ are fed back to the controller and guidance blocks.

Figure 3.1 The control flow of the system



3.2.1 Yaw rate controller

The yaw rate controller is based on section 12.2.9 in Fossen (2011). The yaw dynamics of the ship are given as:

$$(I_z - N_{\dot{r}})\dot{r} - N_r r = \tau_N \tag{3.6}$$

where $(I_z - N_{\dot{r}}) = M_{6,6} > 0$ is a constant from the vessel's inertia matrix given in equation 3.3. $-N_r = D_{6,6} > 0$ is a constant from the vessel's linear damping matrix given in equation 3.5. The following feedback control law is implemented to regulate r to r_d.

$$\tau_n = -N_r r - K_p^r(r - r_d) \tag{3.7}$$

where $K_p^r > 0$ is a design parameter and the r superscript specifies that this is the yawrate controller gains. r_d is the desired yaw rate given by the higher level controller.

3.2.2 Speed controller

The speed controller is based on the first order Nomoto model for surge motion:

$$u(s) = \frac{K}{T + s} n_c(s) \tag{3.8}$$

where u is the forward speed and n_c is the commanded propeller rotation speed. A PI controller is used to regulate the surge speed, u, to the desired speed, u_d. This controller is given by:

$$n_c = K_p^s\left((u_d - u) + K_i^s \int (u_d - u)dt\right) \tag{3.9}$$

where $K_p^s > 0$ and $K_i^s > 0$ are design parameters and the s superscript specifies that this is the speed controller gains. This controller is asymptotically stable for a constant or slowly varying current disturbance.

3.2.3 Thrust allocation

Thrust allocation is the problem of translating a commanded force and moment vector $\boldsymbol{\tau}$ into actual propeller and thruster Revolutions Per Minute (RPM) and angles, $\boldsymbol{u}$.

$$\boldsymbol{\tau} = \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{K}\boldsymbol{u} \tag{3.10}$$

$\boldsymbol{T}(\boldsymbol{\alpha})$ is the thrust configuration matrix which depends on $\boldsymbol{\alpha}$, the azimuth thruster angles. The thrust configuration matrix describes the geometry of the thruster placement on the ship relative to the mass center. $\boldsymbol{K}$ is a diagonal force coefficient matrix.

The thrust allocation problem is solved by the provided Simulink model, and is not considered more in depth. A more thorough discussion is presented in section 12.3 in Fossen (2011).

3.2.4 Actuator saturation and realistic propulsion

In addition to a thrust allocation block, the provided model includes an actuator saturation block and a realistic propulsion block. The actuator saturation block ensures that the commanded force and moment vector $\boldsymbol{\tau}$ is saturated when exceeding actuator capabilities. The realistic propulsion block ensures that the actual propeller and thruster RPM and angles, $\boldsymbol{u}$, translate into realistic forces and moments on the vessel model. Both of these are used like a black box, as they add more realism to the ships motions from the controllers commanded signals.

3.3 Collision avoidance methods

This section discusses different types of collision avoidance methods. First some preliminary terms are introduced. Then a discussion of global and local collision avoidance methods are presented. The methods used for the path planner in chapter 4 and dynamic collision avoidance in chapter 6 are based on the discussions presented here.

3.3.1 Global vs local collision avoidance methods

Global methods of collision avoidance are also called path planning or motion planning methods. They have access to the entire map of their surroundings and of the obstacles in it. They use the map and the information in it to find a path from their initial position to their predefined goal. The goal state can be a set of coordinates as well as a full description of the vehicle state, such as orientation and velocities. Most global methods will find a path as long as there exists a feasible one. Global methods use information that is not necessarily sensable from the vehicle at all times. This means that it cannot account for obstacles that are not part of the environment map and are not suitable for rapidly changing environments. Another drawback of global methods is that they are computationally expensive. Their computation time might range from seconds to minutes. This makes them unsuitable for reactive collision avoidance to dynamic situations.

Local methods are generally reactive algorithms demanding much less computational time than their global counterparts. They cannot guarantee to reach the goal, as only their immediate surroundings are considered which can often lead them into local minima. Local methods generally output commands directly to the motion controller in terms of desired forces or velocities and yaw rates as opposed. These properties make them more suitable for reactive collision avoidance than global methods.

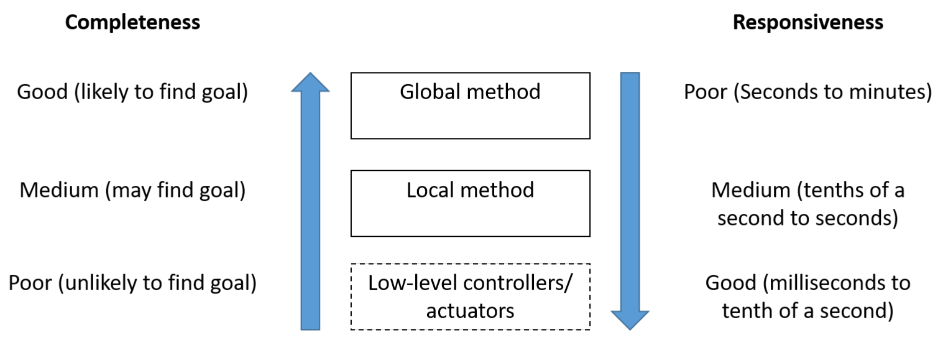
3.3.2 Hybrid methods

Hybrid methods seek to mend the weaknesses of local and global methods by combining them. In this approach, the global method plans the path for the local method to follow. The local method will try to stay on this path but will deviate to avoid dynamic obstacles in its way. The structure of the system is shown in figure 3.2. The higher reaction time of the local method will make the system more robust in avoiding obstacles, while the global method will likely ensure the goal is reached in the end.

3.4 Global methods

The three main global methods from Loe (2007) are briefly presented and evaluated in the context of this thesis based on his results. The choice of algorithm in chapter 4 is based on these methods.

Figure 3.2 The properties a hybrid collision avoidance system.


Completeness
Responsiveness
Good (likely to find goal)
Global method
Poor (Seconds to minutes)
Medium (may find goal)
Local method
Medium (tenths of a second to seconds)
Poor (unlikely to find goal)
Low-level controllers/ actuators
Good (milliseconds to tenth of a second)

3.4.1 Shortest path algorithms

Shortest path algorithms seek to find the shortest path between nodes in a graph. The length of the path found by summing up the cost of the edges of the path.

Dijkstra's algorithm

Dijkstra's algorithm is the classical solution to the one-to-all shortest path problem. This means it finds the shortest path from one node to all the other connected nodes in the graph. Dijkstra's algorithm works on non-negatively weighted and directed graphs. The weights of the edges would in the context of this thesis likely be the Euclidean distance between nodes. The algorithm would stop when the shortest path to the goal node has been found.

A-Star

The A* algorithm is a global and optimal method for finding the shortest path between two points. It is complete, meaning it will always find a path if it exists. The A* algorithm is a combination of Dijkstra's algorithm and heuristics with which it achieves computational optimality [Wikipedia (2017)]. The heuristics function $h(x)$ helps decide which node to explore next in the queue. It needs to be admissible for A* to find a minimal cost path. To be admissible it must never over-estimate the cost to the goal.

A* uses a "best-first" approach, meaning that there may be many equally good paths, but it will only select the first one it finds. To make use either algorithm, the environment map must be decomposed into a graph of connected nodes. The easiest way of doing this is to construct a bitmap of evenly sized rectangles, where a 1 at location (i, j) symbolizes an obstacle, and 0 symbolizes a free space at (i, j). This map can be expanded in more dimensions to include height or time information etc.

D-Star (D*)

The D* algorithm known as the dynamic A* algorithm is a variant of the A* algorithm which can more easily deal with a varying environment. It has the ability to recover its path to a degree when the environment has changed. It may be more efficient than A* in a dynamic environment.

Evaluation

The A* algorithm is an improved variant of Dijkstra's algorithm for the context of this thesis. It can be extended to weight edge connections based on proximity to obstacles, and change of path angle. This will create paths with smooth curvature and a safe distance away from obstacles. A* will create the optimal shortest path as long as a solution exists, but run-time may be high. Its main weakness is its inability to include vessel dynamics in the path planning, which may result in paths the vessel is unable to follow.

3.4.2 Rapidly-exploring random trees

The RRT method is a path planning method capable of taking the dynamics of the vessel into account. It is a randomized method and as such can explore most of the space of possible solutions very quickly compared to a complete method. The solution is however not optimal but is usually good enough.

The RRT method works by creating a tree of examined nodes, initially only containing the starting position of the craft. It begins by examining a random state. For this state it finds the nearest neighbor in the tree by some metric (for example the Euclidean distance) then it tries to connect the two states using a motion planner. If successful the state examined is added is connected as a child to its neighbor. The method takes into account the dynamics of the vessel in the form of the motion planner. Eventually, the goal state will be connected to the tree and a path will be found. The series of inputs made by the motion planner is saved.

Evaluation

An important advantage of the RRT method is that it outputs the whole vessel state, including heading, velocity, and position along the path. A good motion planner ensures that the path is feasible. The method is reasonably fast and is simple to modify and extend. Possible extensions are changing the nearest neighbor metric to favor time, fuel usage, path length, etc. The method can also be extended to take dynamic objects into account.

The main disadvantages of the method are complexity of implementation and the path sub-optimality. The paths generated tend not to be straight, making unnecessary bends in open areas.

3.4.3 Constrained nonlinear optimization

The usage of constrained nonlinear optimization to control dynamic systems is often called Model Predictive Control (MPC). A general nonlinear optimization problem can be stated as:

$$
\begin{aligned}
\min_{\mathbf{X}\in\mathbb{R}^n} \quad & J(\mathbf{X}) \\
\text{s.t.} \quad & \\
& c(\mathbf{X}) \leq 0 \\
& ceq(\mathbf{X}) = 0
\end{aligned}
\tag{3.11}
$$

where $J : \mathbb{R}^n \rightarrow \mathbb{R}$ is a continuous, smooth function with a well-defined gradient and Hessian. The purpose of the method is to minimize J while still satisfying the inequality and equality constraints $c(x) \leq 0$ and $ceq(x) = 0$ respectively. Several algorithms for solving such a problem exist and a more detailed discussion of these kinds of problems is presented in Nocedal and Wright (2006).

For a dynamic system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ one way of formulating the path planning problem is to define the solution of the optimization problem to be a sequence of inputs $\mathbf{u}_i$. This sequence should bring the system to its goal $\mathbf{x}_f$ while avoiding collisions.

$$
\mathbf{X} = [u_1 \; \delta t_1 \; u_2 \; \delta t_2 \; \cdots \; u_n \; \delta t_n] \tag{3.12}
$$

The MPC approach then seeks to solve the following problem

$$
\begin{aligned}
\min_{\mathbf{X}\in\mathbb{R}^n} \quad & J(\mathbf{X}, \mathbf{x}(\cdot)) \\
\text{s.t.} \quad & \\
& h(\mathbf{x}(\cdot)) \leq 0 \\
& \mathbf{x}_f - \mathbf{x}(t_f) = 0
\end{aligned}
\tag{3.13}
$$

One of the constraints is that the final position of the model is the same as the goal position. The other constraint, $h(\mathbf{x})$, can be added to represent obstacles etc. The state of the vessel at each time epoch is found by integrating the dynamic model $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ from its initial position, $\mathbf{x}(0) = \mathbf{x}_0$.

The selection of the cost function J determines the qualities the generated path is optimized for, such as path length, time of path or fuel consumed.

Evaluation

The method may seem the ideal method for the problem of this thesis. It generates an optimal and feasible path as long as the model is good. It has its problems, however. The problem in non-convex so a global solution cannot be guaranteed, meaning that the solution is not necessarily optimal. The method might even not be able to generate a path at all if the environment is cluttered. Using another method to generate an initial solution will solve this. The computational power required is quite large for this method compared to the other two.

3.5 Local methods

Three local methods are presented with varying degrees of complexity. Local methods are used mostly for dynamic obstacle avoidance. Chapter 6 is based on the results discussed here. A simple introduction and evaluation of each method is presented. A full review is presented by Loe (2007).

3.5.1 Dynamic window

The dynamic window approach is designed to take the limitations of vehicle velocities and turn rates into account. This ensures that the method outputs only feasible control outputs, which is critical for dynamic obstacle avoidance. The DW method assumes a constant velocity and turn-rate over a given time period. The vessel trajectory can then be estimated as a straight line or a constant radius arc.

The search space of the algorithm at time interval i is the possible translational and angular velocities (u_i, r_i) respectively. The algorithm must choose the best pair of these. First, the search space must be restricted down to only allow certain pairs of velocities. The first restriction is to only allow velocities which will not place the vessel in danger during the next time interval. This means that the vessel must be able to come to a complete stop during the next interval. This is called the *admissible velocities*.

The second restriction is to only allow velocities that can be reached in the next time interval. This represents the limitations in vessel acceleration. Selecting the optimal pair of velocities is done by maximizing an objective function with the vessel velocities as inputs. The function is a linear combination of heading, distance, and velocity of the arc. This ensures a fast and short path is chosen which brings the ship towards its target heading.

Evaluation

The main benefit of this method is its ability to provide feasible outputs taking vessel dynamics into account. Its computational requirement is higher than the other methods discussed below.

3.5.2 Artificial potential field (APF)

The artificial potential field is an intuitive and simple method. It is based on attractive and repulsive forces. The method works having the goal apply an attractive force on the vessel while obstacles apply a repulsive force. The sum of these forces is supplied to the motion controller/thrust allocation of the vessel. The repulsive force of an obstacle is taken from the closest point to the vessel and is inversely proportional to the distance from the ship. At a certain distance away no force is applied. The attractive force is proportional to the distance between the vessel and the goal. To ensure consistent behavior the attractive force is given an upper limit.

Evaluation

The method is prone to get trapped in local minima when the repulsive forces cancel the attractive ones. This is made more likely the more obstacles are present. These local minima may cause the method to be unable to guide the vessel through narrow passages. The method is also prone to oscillations near obstacles or in narrow passages. Similar methods and improvements to this method exist which may help to solve the local minima issues to an extent.

The main issue though is the fact that the output is a desired force. This is a problem for underactuated ships where the desired force may cause motion in an unwanted direction. This method is therefore much better suited for highly controllable, holonomic systems.

3.5.3 Vector field histogram (VFH)

The vector field histogram method is designed to improve on the APF method by removing the oscillating behavior near obstacles. It uses a *Cartesian histogram grid*, C, to store information about obstacles in the environment. The cells of C contain the probability of the cell containing an obstacle. The APF method would use this map directly to generate the potential fields. The VFH method will use an intermediate world-representation to make better control decisions. All cells outside a given radius are ignored to save computational power.

A polar histogram, H, is generated from the restricted C. This one-dimensional histogram consists of the angular sectors around the vessel position. Each bin of the histogram represents the density of obstacles in that sector. In effect, the high points of H represent which directions from the vessel position there are obstacles. The low points of H represent directions where the path is clear.

The selection of the next steering command is done by examining all the valleys of H and selecting the one which is closest to the goal heading. The velocity commanded is a function of the density of obstacles in the current direction of travel.

Evaluation

This method does remove the oscillations experienced by the APF method and the issues with navigating narrow passages. It does not, however, remove the problem of local minima. It also doesn't take vehicle dynamics into account which may lead it to demand impossible controls.

Chapter 4

Path planner

The A* is the chosen method for solving the path planning problem. The A* algorithm has been extensively used in large-scale navigation problems as part of the path planner. An example of this can be found in Larson et al. (2006). Because it does cost analysis at each step it is possible to process the map to incur costs on variables such as proximity to obstacles, direction, shipping lanes, "soft" obstacles, route time, etc [Larson et al. (2006)]. A drawback of A* is that it does not include vessel dynamics in the process. This must be compensated for by the other parts of the path planner and follower.

Compared to the RRT it outputs a better path since its solution is optimal. The RRT methods advantage of being able to optimize between fuel usage, time and path length considers only the optimality of each segment, not the whole path. Therefore the A* method may in total perform better than RRT in these regards without optimizing for it. Weighting based on the change of path angle is not as robust as the RRT path in the sense of including vessel dynamics. With correct tuning, however, it may be sufficient.

Compared to constrained nonlinear optimization A* is much simpler to implement while still generating an optimal path. Constrained nonlinear optimization is also significantly more time consuming to compute.

4.1 Implementation of A*

The algorithm systematically explores the graph/map from the given start coordinates using an *open* and a *closed* set as shown in Algorithm 1. The *open* set contains all discovered nodes that are yet to be examined. The *closed* set contains the fully processed nodes. The heuristic estimate from the node to the goal is called $h(x)$, and the cost of the path from start to the node is called $g(x)$. At each iteration the algorithm finds the node with the lowest estimated cost, $h(x) + g(x)$, from the *open* set. Then it examines all its neighbors, finding their total cost $g(x)$ and adding them to the open set. If the currently processed node happens to be the given goal node then the path has been found and the algorithm is finished. The obstacles are usually represented as a bitmap and are added to the *closed* set so that their locations are inaccessible to the algorithm.

Algorithm 1 A*

```
1: $closed \leftarrow \emptyset$
2: $open \leftarrow$ start
3: $cost(start) \leftarrow 0$
4: while open $\neq \emptyset$ do
5:     $node \leftarrow$ EXTRACT_MIN($open$)
6:     if $node$ = $goal$ then return GET_PATH($node$)
7:     for $nb \leftarrow$ GET_NEIGHBORS($node$) do
8:         cost $\leftarrow cost(node) +$ MOVECOST($node, nb$)
9:         if $(nb \in open) \wedge (cost \geq cost(open(nb)))$ then
10:            go to 7
11:        if $(nb \in closed)$ then
12:            go to 7
13:        $parent(nb) \leftarrow node$
14:        $cost(nb) \leftarrow cost$
15:        $open \leftarrow nb$
16:    $closed \leftarrow node$
```

4.2 Connecting distance

The set of neighbors for each node is usually represented by only the four adjacent cells, resulting in a search on only the cardinal directions (north, south, east and west) around the node. The *connecting distance* can be increased which increases the set of neighbors to include the eight adjacent cells, or even more as shown in [Ueland et al. (2017)]. This generates a smoother path as more path orientations are considered as shown in in figure 4.1. The line from the node to each neighbor must not be allowed to cross any obstacle. A drawback of increasing the connecting distance is that the computation time increases significantly for high values.

4.3 Penalizing closeness to obstacles

The map of the environment provided to the algorithm is weighted in order to incite the algorithm to keep a little distance from obstacles. This prevents it from planning the path as tightly as possible around obstacles. One weighting scheme is presented in [Ueland et al. (2017)] as follows:

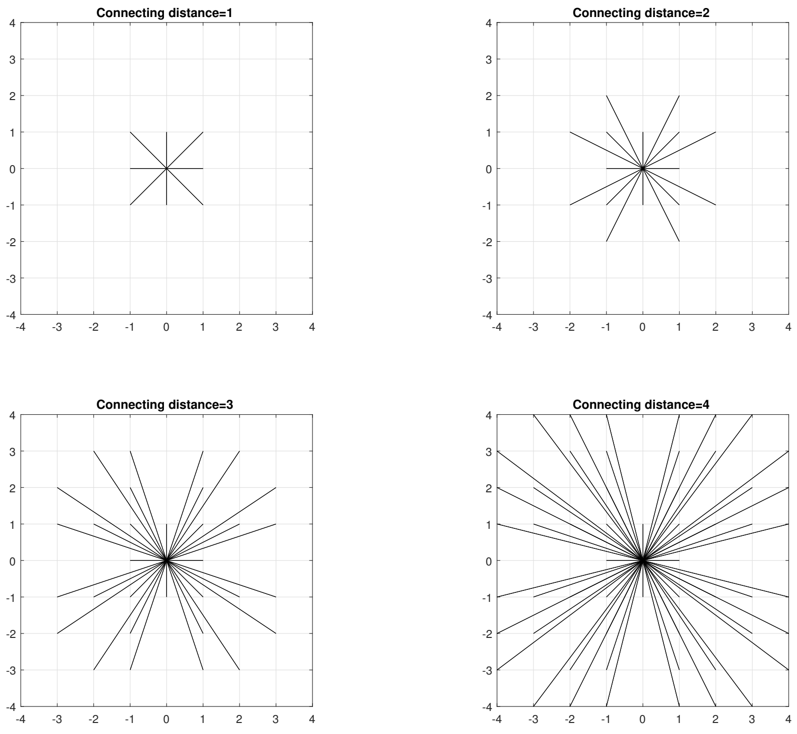
$$w_d(i,j) = 1 + \frac{n}{p + d_{obj}} \tag{4.1}$$

where d_{obj} is the Euclidean distance to the closest obstacle from node (i,j). n and p are tuneable parameters to suit the vessel dynamics and objective. Figure 4.2a shows a weighted cost map where the obstacles are represented in black.

Another weighting scheme is proposed as:

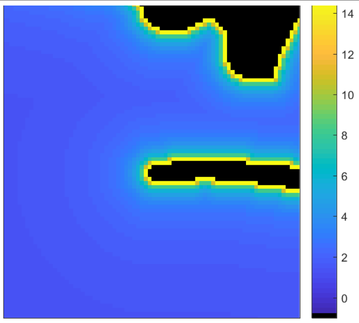
$$w_d(i,j) = ae^{-bd_{obj}} \tag{4.2}$$

Figure 4.1 Discoverable neighbors from a node for increasing connecting distances

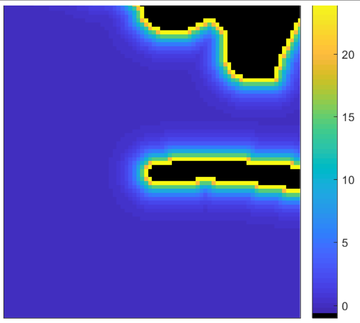


This tends to zero much faster with increasing distance than the previously stated equation 4.1. This is shown in figure 4.2b.

The weight of each node is added to the total path cost $g(x)$. In the case of connecting distances larger than 1, there is a problem if the line from the current node to the neighbor passes close to an obstacle but the neighbor far away from any obstacle. Therefore the weights of all the nodes crossed by the line are checked and the highest one is used.



(a) Weighted cost map of the environment based on equation 4.1.



(b) Weighted cost map of the environment based on equation 4.2.

4.4 Penalizing sharp turns

To incite the algorithm to choose paths that are feasible a penalty cost is incurred for a large change of angle between two nodes. The cost is given in equation 4.3, where r is a tunable parameter. A and B are the vectors from the parent node to the current node, and from the current node to the neighbor in question respectively.

$$cos(\theta) = \frac{A \cdot B}{|A||B|} \tag{4.3a}$$

$$w_\theta(i,j) = r\theta^2 \tag{4.3b}$$

4.5 Evaluation of A*

The A* method guarantees to find the shortest path as long as it exists. The existence of a path may depend on the resolution of the environment map. It uses limited computational power but may use large amounts of memory [Loe (2007)]. This is not of concern for the implementation on a large ship as the cost and weight of the required hardware is small compared to the rest of the operation.

A simple implementation of the A* will result in jagged paths. The vessel dynamics may cause these paths be infeasible when only using the stern propeller. The proposed augmentations mentioned above make the path smooth and it will favor large, feasible turns.

Should it be unable to construct a path without sharp turns, the ship has access to azimuth thrusters and is able to complete the path using dynamic positioning. This situation is of course not efficient, so it important to tune the curve minimization to suit the dynamics of the underactuated ship.

Chapter 5

Path following controller

The path following controller must be chosen to suit the parameterization path it is meant to follow. The path generated from chapter 4 is a smooth parameterized path. Therefore the *path following kinematic controller* from Fossen (2011) is chosen based on the discussion in 2.3. This controller is designed to follow smooth parameterized paths.

5.1 The Serret-Frenet frame

The path following kinematic controller works by tracking a virtual target on the path. In order to generate the error states for the controller a reference frame that moves along the path is needed. The most commonly used reference frame is the SF frame. The virtual target is then defined as the projection of a vessel on a path tangential reference frame.

The SF frame is depicted in figure 5.1. The *cross track error*, e, represents the distance from the craft from the path tangent. It can be seen as the deviation from the path for smaller values of s. The *along track error*, s, is the trailing distance of the ship behind the virtual target, and is used as a design parameter. When path following the ship travels in cruise speed without temporal constraints, the virtual target is moved along the path at a rate to keep s constant. The SF course, χ_{SF}, is shown in figure 5.2 and is defined as the angle between the SF x-axis, $\vec{x}_{SF}$, and the ship's speed vector, $\vec{U}$.

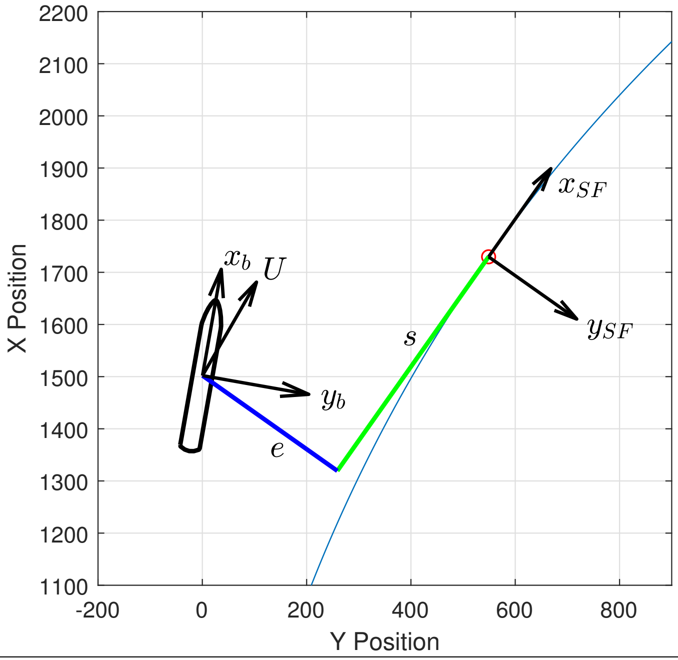
5.2 The path following kinematic controller

The error states for the controller are e, s and $\tilde{\chi}_{SF} = \chi_{SF} - \chi_d$, and the goal of the controller is to drive these to zero. This will align the body frame of the ship with the path tangential SF frame. The desired approach angle, χ_d, is chosen as follows:

$$\chi_d(e) = \arctan\left(\frac{-e}{\Delta}\right) \tag{5.1}$$

This is the angle of the line of sight vector to a point on x_{SF} located a lookahead distance ahead. Figure 5.2 shows the desired approach angle and the lookahead distance. This

Figure 5.1 Description of the SF frame



steering scheme will be feasible for any cross track error. A longer lookahead distance will yield a slower and more gentle approach to the path, while a shorter one will be more aggressive, which can lead to oscillations.

The path following kinematic controller is derived in section 10.4.2 in Fossen (2011). It is given as:

$$r_d = \left(1 - \frac{(m - X_{\dot{u}})}{(m - Y_{\dot{v}})}\right)^{-1} \left[\dot{\chi}_d + \kappa U_d - K_1 \tilde{\chi}_{SF} - \frac{Y_v}{(m - Y_{\dot{v}})}\left(\tan(\beta) - \frac{v_c}{U\cos(\beta)}\right)\right] \tag{5.2}$$

$$U_d = U\cos(\chi_{SF}) + K_2 s \tag{5.3}$$

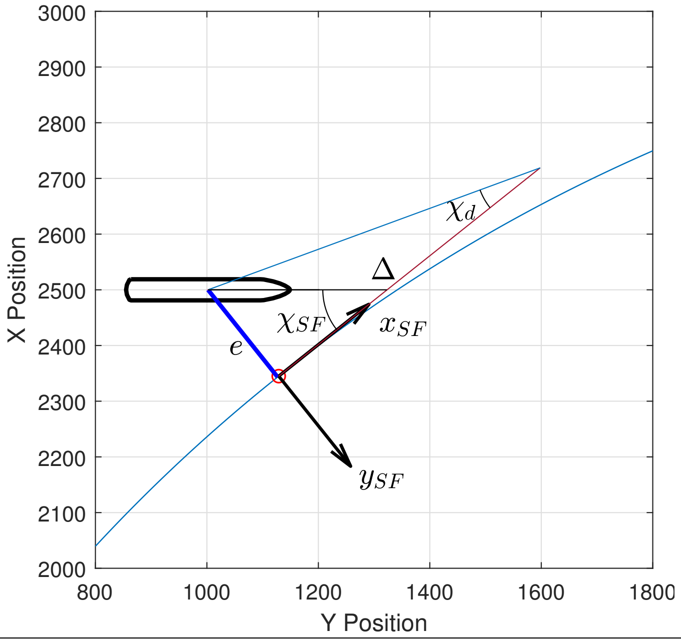
where r_d is the desired yaw rate and U_d is the desired path-tangential speed. The sideslip angle, $\beta = \arcsin\left(\frac{v}{U}\right)$, and the current velocity, v_c, must be measured or estimated in a state observer. In the simulations they are available and used as such. For very low to zero speed the $-\frac{Y_v}{(m - Y_{\dot{v}})}\left(\tan(\beta) - \frac{v_c}{U\cos(\beta)}\right)$ term is disabled and set to zero.

The path curvature $\kappa(\omega)$ at the location of the virtual target is given by:

$$\kappa(\omega) = \frac{x'_d y''_d - y'_d x''_d}{\left((x'_d)^2 + (y'_d)^2\right)^{(3/2)}} \tag{5.4}$$

The path provided by the path planner is a series of finely spaced coordinates in the North-East-Down (NED) frame. To generate x'_d, y'_d, x''_d and y''_d the path must be numerically

Figure 5.2 LOS based steering with lookahead distance Δ



differentiated. Numeric differentiation generates significant noise in the signal. The curvature function $\kappa(\omega)$ is smoothed by a 1-d normalized Gaussian filter in order to ensure a smooth change of curvature along the path. The size of the filter window must be fitted to the fineness of the path interpolation.

5.3 Adjustments to the controller

In practice, only the desired yawrate, r_d, from the controller is used. The desired speed is always the cruising speed, and hence U_d is unnecessary. As stated earlier, s is used as a constant design parameter. By setting $s = 0$, the virtual target will always be on the point of the path closest to the ship. This reduces the control objective to only regulate the cross track error, e, and the SF course error, χ_{SF}. The virtual target is designed to always move to ensure that s is kept constant.

Chapter 6

Dynamic collision avoidance

The dynamic collision avoidance method chosen is based on Barisic (2012). The VPM method presented here is an APF method as discussed in section 3.5.2. The weaknesses of APF methods are local minima, difficulties with narrow passages, and the inability of the ship to follow an arbitrary desired force vector. The VPM has accounted for the weaknesses of local minima and narrow passages by introducing a rotor field around obstacles in addition to the repulsive field. Modifications presented in this chapter reduce the difficulties with following the force vector.

6.1 The virtual potential method

The *final virtual potential function* is defined as the finite sum:

$$P_\Sigma = \sum_{i=1}^{n} P_i \tag{6.1}$$

where P_i is the virtual potential of the i-th component. A component can be either an obstacle possessing a repulsive potential field or a waypoint possessing an attractive potential field. The *decentralized total control function* is then defined as:

$$\forall \mathbf{x} \in \mathbb{C} \subseteq \mathbb{R}^2, \mathbf{E}(\mathbf{x}) = -\nabla P_\Sigma(\mathbf{x}) \tag{6.2}$$

where $\mathbb{C}$ is the navigable waterspace; the connected subset of of $\mathbb{R}^2$ which excludes all obstacles. $\mathbf{E}(\mathbb{C}) : \mathbb{C} \to \mathbb{R}^2$ is the real-valued 2d vector field over $\mathbb{C}$ consisting of the commanded ideal accelerations for any point in $\mathbb{C}$. This field does not take into account the holomomic constraints or the dynamic model of the ship.

The *ideal conservative trajectory*, which is the trajectory given when ideally following the decentralized total control function, is given by:

$$\mathbf{x} = \int_0^t \int_{\tau=0}^t \mathbf{E}[\mathbf{x}(\tau)] d\tau dt + \mathbf{x}_0 \tag{6.3}$$

By its design, this trajectory is not guaranteed to be convergent to the goal, or free of local minima. A discussion on the passivity and the local minima in the context of the virtual potential method for motion planning is given in (Barisic, Vukic and Miskovic; 2007a), (Barisic, Vukic and Miskovic; 2007b) and (Barisic, Vukic, Miskovic and Tovornik; 2007).

6.2 Potential functions

There are two types potential functions that are summed together in the final virtual potential function, the *obstacle potential function* and the *waypoint potential function*. For the sake of the motion planning problem, the requirements on these are:

1. The potential function of obstacles, $P_o(d)$, decreases monotonously with increasing distance d. The acceleration is always directed away from the obstacle.
2. The acceleration from the obstacle will tend to ∞ as d goes to zero. The acceleration tends to 0 as d increases.
3. The potential function of waypoints, $P_w(d)$, increases monotonously with increasing distance d. The acceleration is always directed towards the waypoint.
4. The acceleration towards the waypoint will decrease linearly to zero with decreasing distance to the waypoint. The acceleration will be limited/saturated at a certain distance d_0 from the waypoint, causing a constant acceleration at any distance greater than d_0.

The potential function for an obstacle used is given below and shown in figure 6.1a:

$$P_o(d) = e^{\left(\frac{A_o}{d}\right)} - 1 \tag{6.4}$$

$$\frac{\partial}{\partial d} P_o(d) = -\frac{A_o}{d^2} e^{\left(\frac{A_o}{d}\right)} \tag{6.5}$$

$$\lim_{d\to\infty} P_o(d) = 0 \tag{6.6}$$

$$\lim_{d\to 0^+} P_o(d) = \infty \tag{6.7}$$

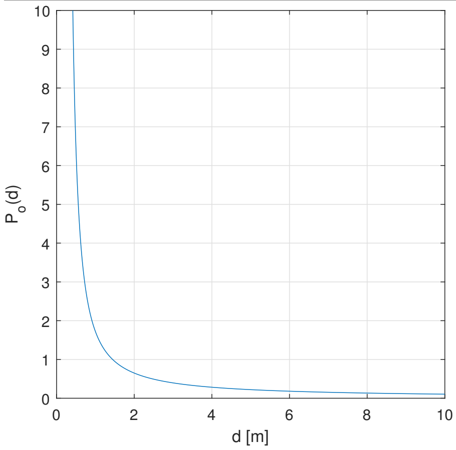
$$\lim_{d\to\infty} \frac{\partial}{\partial d} P_o(d) = 0 \tag{6.8}$$

$$\lim_{d\to 0^+} \frac{\partial}{\partial d} P_o(d) = \infty \tag{6.9}$$

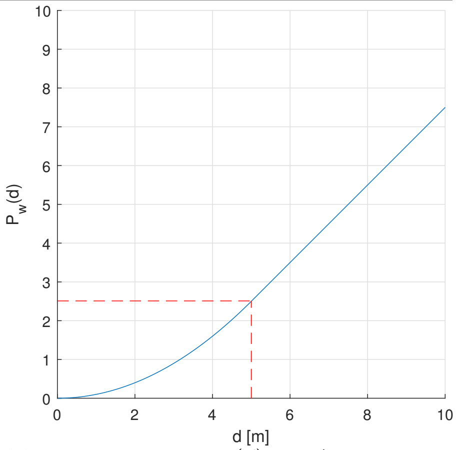
The potential function for a waypoint used is given below and shown in figure 6.1b:

$$P_w(d) = \begin{cases} \frac{1}{2}\frac{A_{w0}}{d_{w0}} d^2, & \text{if } d \le d_{w0} \\ A_{w0}(d - d_{w0}) + \frac{A_{w0} d_{w0}}{2}, & \text{otherwise} \end{cases} \tag{6.10}$$

$$\frac{\partial}{\partial d} P_w(d) = \min\left(\frac{A_{w0}}{d_{w0}} d, A_{w0}\right) \tag{6.11}$$



(a) Graph showing $P_o(d)$ for $A_o = 1$



(b) Graph showing $P_w(d)$ for $A_{w0} = 1$ and $d_{w0} = 5$

The decentralized control function, $\boldsymbol{a}$, is then given as:

$$\boldsymbol{a} = -\nabla P_\Sigma(\boldsymbol{x}) = -\nabla \sum_i P_i(\boldsymbol{x}) \tag{6.12}$$

$$= \sum_i \Big(-\nabla p_i(d_i(\boldsymbol{x}))\Big) \tag{6.13}$$

$$= -\sum_i \frac{\partial}{\partial d_i(\boldsymbol{x})} p_i(d_i(\boldsymbol{x})) \cdot \hat{\boldsymbol{n}}_i(\boldsymbol{x}) \tag{6.14}$$

where i is the index of all obstacles and waypoints.

6.3 Rotor function

As stated above, artificial potential field methods are very susceptible to local minima. In order to address this problem an additional decentralized control function, called the *rotor decentralized control function* is added to each obstacle. These function similarly to the repulsive potential functions of obstacles, only they direct their force perpendicular to the normal of the obstacle. The potential field is directed either clockwise or counter-clockwise depending on which way is shorter from the ship to the waypoint. Such a rotor function will force any approaching object to travel around the obstacle when it gets close. To prevent getting stuck on the obstacle, the rotor function is zero when the ship has passed the obstacle. The rotor decentralized control function, $\boldsymbol{a}^{(r)}$ is given by

$$\boldsymbol{a}_i^{(r)} = -\frac{A_r}{d_i(\boldsymbol{x})^2} e^{\left(\frac{A_r}{d_i(\boldsymbol{x})}\right)} \hat{\boldsymbol{a}}_i^{(r)}(\boldsymbol{x}) \tag{6.15}$$

$$\hat{\boldsymbol{a}}_i^{(r)}(\boldsymbol{x}) = \hat{\boldsymbol{r}}_i^{(r)}(\boldsymbol{x}) \times \begin{bmatrix}\hat{\boldsymbol{n}}_i^{(r)}(\boldsymbol{x}) & | & 0\end{bmatrix}^T \tag{6.16}$$

$$r_i(\boldsymbol{x}) = \begin{bmatrix}\frac{\boldsymbol{w}_k - \boldsymbol{x}_i}{||\boldsymbol{w}_k - \boldsymbol{x}_i||} & | & 0\end{bmatrix}^T \cdot \begin{bmatrix}\hat{\boldsymbol{n}}_i^{(r)}(\boldsymbol{x}) & | & 0\end{bmatrix}^T \tag{6.17}$$

$$\hat{\boldsymbol{r}}_i^{(r)}(\boldsymbol{x}) = \begin{cases} r_i = -1 & \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T \\ -1 < r_i \leq 0 : & \text{sign}\left(\begin{bmatrix}\frac{\boldsymbol{w}_k - \boldsymbol{x}_i}{||\boldsymbol{w}_k - \boldsymbol{x}_i||} & | & 0\end{bmatrix}^T \times \begin{bmatrix}\hat{\boldsymbol{n}}_i^{(r)}(\boldsymbol{x}) & | & 0\end{bmatrix}^T \right) \\ \text{otherwise :} & \vec{0} \end{cases} \tag{6.18}$$

where:

- A_r is a tunable design parameter similar to A_o and A_w, determining the amplitude of acceleration.
- $\hat{\boldsymbol{a}}_i^{(r)}(\boldsymbol{x})$ is the unit direction vector of the rotor decentralized control function
- $\hat{\boldsymbol{n}}_i^{(r)}(\boldsymbol{x})$ is the unit outwards normal to the obstacle centre.
- $r_i(\boldsymbol{x})$ is the rotor direction discriminator.
- $\hat{\boldsymbol{r}}_i^{(r)}(\boldsymbol{x})$ is the unit rotation direction generator. When $r_i(\boldsymbol{x}) \leq 0$ it means that the obstacle is in front of the ship with respect to the waypoint, so $\hat{\boldsymbol{a}}_i(\boldsymbol{x}) \neq 0$. When $r_i(\boldsymbol{x}) > 0$ it means that the obstacle is behind the ship with respect to the waypoint, so $\hat{\boldsymbol{a}}_i(\boldsymbol{x}) = 0$. When $r_i(\boldsymbol{x}) = -1$ an arbitrary rotation direction is chosen.

6.4 Derivation of control signals

The motion planning solution derived from the decentralized total control function $\mathbf{f}(k) = \mathbf{E}(\mathbf{x})$ is not guaranteed to satisfy the constraints of the control problem. These constraints are given by the dynamics of the ship and it's operational limits in thrust and torque. This section presents the method of translating the solution of the decentralized total control function into low level control signals for the speed and yaw controllers.

The control signals are generated using simple Euler backwards integration with a sample time of T:

$$\begin{aligned}
u_c(k) &\leftarrow ||T \begin{bmatrix} f_u(k) & f_v(k) \end{bmatrix}^T + \begin{bmatrix} u(k-1) & 0 \end{bmatrix}^T || && (6.19)\\
&= \sqrt{T^2 f_u(k)^2 + 2T f_u(k) u(k-1) + u(k-1)^2 + T^2 f_v(k)^2} && (6.20)\\
\dot{u}_c(k) &\leftarrow \frac{u_c(k) - u_c(k-1)}{T} && (6.21)\\
\psi_c(k) &\leftarrow \text{atan2}\big(T f_v(k),\ u(k-1) + T f_u(k)\big) + \psi(k-1) && (6.22)\\
r_c(k) &\leftarrow \frac{1}{T}\text{atan2}\big(T f_v(k),\ u(k-1) + T f_u(k)\big) && (6.23)\\
\dot{r}_c(k) &\leftarrow \frac{r_c(k) - r_c(k-1)}{T} && (6.24)
\end{aligned}$$

where atan2(y, x) is the four quadrant inverse tangent function. The expression for the forces are given in the body system, $T_b^{-1}\mathbf{f}(k) = \mathbf{f}_b(k) = \begin{bmatrix} f_u(k) & f_v(k) \end{bmatrix}$.

The control signals u_c and r_c are limited to the system to satisfy the following constraints:

$$\begin{aligned}
u_c(k) &= \text{sign}(u_c(k))\text{min}(|u_c(k)|,\ V_{max}) && (6.25)\\
\dot{u}_c(k) &= \text{sign}(\dot{u}_c(k))\text{min}(|\dot{u}_c(k)|,\ A_{max}) && (6.26)\\
r_c(k) &= \text{sign}(r_c(k))\text{min}(|r_c(k)|,\ \omega_{max}) && (6.27)\\
\dot{r}_c(k) &= \text{sign}(\dot{r}_c(k))\text{min}(|\dot{r}_c(k)|,\ \alpha_{max}) && (6.28)
\end{aligned}$$

where $V_{max})$ is the maximum forward speed, $A_{max})$ is the maximum acceleration, $\omega_{max})$ is the maximum yawrate and $\alpha_{max})$ is the maximum yaw-acceleration. The values of these limits must be based on the dynamics of the ship. This will ensure that the controller doesn't command infeasible control signals.

6.5 World representation

The obstacles in the simulation are represented in two different spaces; the *global space*, and the *local space*. The global space represents every obstacle in the simulation and is the ground truth of their positions, shapes, and orientations. The local space is the subset of the global space which is visible/sensable from the ship. The current waypoint is always present in the local space.

The purpose of the local space is to simulate the real world problem of detecting obstacles and locating them on the ship's internal map, similar to the SLAM problem. The local space is defined as all obstacles within the circle of the ship's *sensor range*. Figure 6.2a shows all the obstacles in the local space as red, and all outside as black. The green circle represents the sensor range.

6.6 Augmentations to the VPM

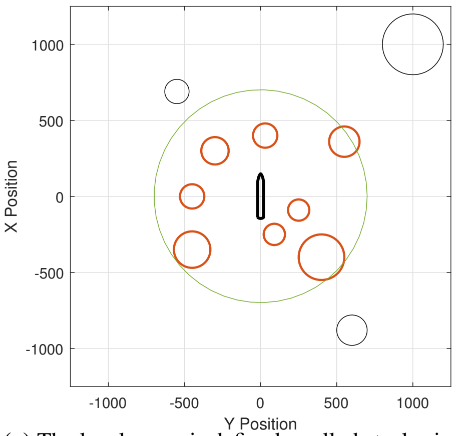
In this thesis only the commanded yawrate r_c is used for dynamic collision avoidance. The ship will always be travelling at cruise speed in this scenario, so u_c is disregarded.

The waypoint is set to be the point on the path where the virtual target has an along track error s equal to the sensor range. This method ensures that the method has a waypoint for any distance between the ship and the path. This way the VPM method will always seek to follow the path, but it will avoid collisions while doing so.

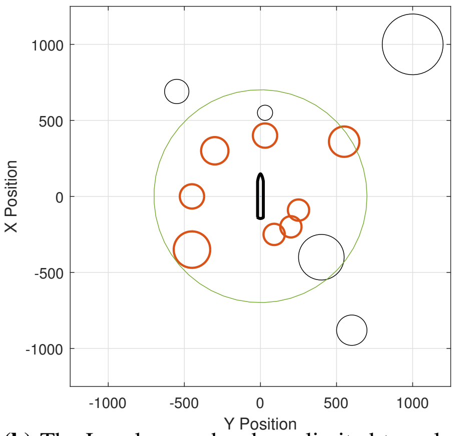
In order to increase performance of the VPM method as a dynamic collision avoidance method, the local space is pruned further. All obstacles in that are completely shadowed by nearer obstacles are pruned away from the local space. This can be seen in the difference between figure 6.2a and 6.2b.

Currently, when passing close to an obstacle close to the path, it exerts its maximum repulsive force when it is directly port or starboard of the ship. However, at this point the ship has already passed the obstacle and does not care about it anymore. The ship should only care about obstacles in front. If it sees that it can pass an obstacle with a safe margin by continuing at its current course angle the obstacle should be disregarded. Therefore, any obstacle outside the field of view of the ship is disregarded as shown in figure 6.2c.

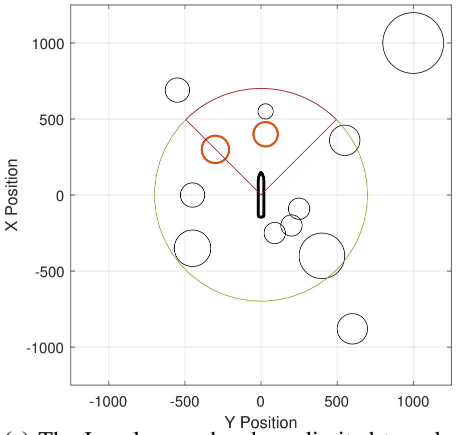
The dynamic collision avoidance system will only be active if there are obstacles in the local space. If there are not, then the path following kinematic controller is used.



(a) The local space is defined as all obstacles in the ship's sensor range (green circle). The red circles are obstacles in the local space, black and red circles are obstacles in the global space.



(b) The Local space has been limited to only include obstacles which are not shadowed by other obstacles.



(c) The Local space has been limited to only include obstacles in the ship's limited field of view. In this example the field of view is $\pm 45°$

Chapter 7

Berthing

The final approach to berth starts when the ship has navigated safely to a predefined point close to its preallocated berthing spot denoted as P_b^i, the initial berthing point. The berthing stage goal is to move from P_b^i to the final berthing point, P_b^f, next to the quay. The challenge in this problem is ensuring the heading of the ship is kept parallel to the quay to avoid collision along the ship's length.

The ships capabilities of full maneuverability allows this problem to be solved as a DP problem. By issuing a slowly changing reference from P_b^i to P_b^f, the DP controller can move the ship in any direction and with any heading.

As discussed in section 2.1 a simple PID DP controller has been implemented to show proof of concept.

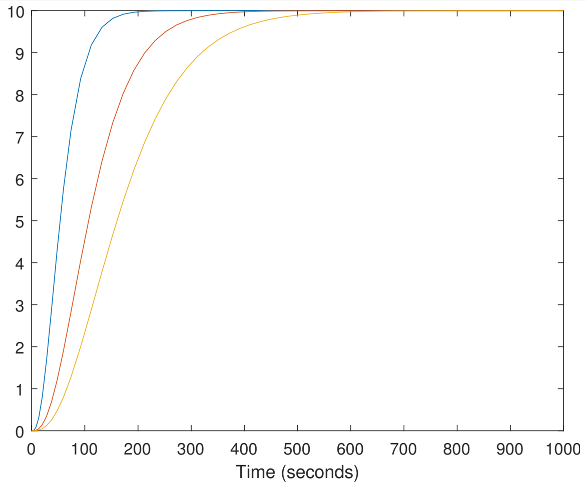
7.1 Reference model

In order to generate a smooth trajectory from P_b^i to P_b^f a reference model is used. The reference model is physically motivated by a mass-damper-spring system which represents the dynamics of the vessel suitably. For position and attitude reference models a third order filter is most often used, in order to generate smooth reference signals in position, velocity and acceleration. This is in effect a first order Low Pass (LP) filter in cascade with the mass-damper-spring system. The reference model is given by:

$$\frac{\eta_{d_i}}{r_i^n}(s) = \frac{\omega_{n_i}^3}{s^3 + (2\zeta + 1)\omega_{n_i} s^2 + (2\zeta + 1)\omega_{n_i}^2 s + \omega_{n_i}^3}, \quad (i = 1, ..., n) \tag{7.1}$$

where $\boldsymbol{r}^n$ the reference vector defined as $P_b^f - P_b^i$ in the $\{n\}$ frame. $\zeta_i (i = 1, ..., n)$ are the *relative damping ratios* and ω_{n_i} are the natural frequencies. These are design parameters to tune the response of the reference model. Choosing $\zeta = 1$ will give a critically damped response. The response of the reference model is shown in figure 7.1. The natural frequencies need to be tuned to the size of the step.

Figure 7.1 Reference model for $\zeta = 1, \omega_{n_i} = 20, 40, 60$



7.2 DP controller

The DP controller is developed in section 12.2.10 in Fossen (2011). The linearized DP controller model is given in vessel parallel coordinates:

$$\dot{\boldsymbol{\eta}}_p = \boldsymbol{\nu} \tag{7.2}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{b}_p + \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave} \tag{7.3}$$

where $\dot{\boldsymbol{\eta}}_p$ are the vessel parallel coordinates which can be related to NED coordinates by:

$$\boldsymbol{\eta} = \boldsymbol{R}(\psi)\dot{\boldsymbol{\eta}}_p \tag{7.4}$$

The bias term $\boldsymbol{b_p} = \boldsymbol{R}(\psi)^T\boldsymbol{b}$ accounts for drift due to waves and currents, as well as unmodeled dynamics

The control system is designed as a MIMO nonlinear PID controller with wind feedforward:

$$\boldsymbol{\tau} = -\hat{\boldsymbol{\tau}}_{wind} - \boldsymbol{R}^T(\boldsymbol{\eta})\boldsymbol{K}_p\tilde{\boldsymbol{\eta}} - \underbrace{\boldsymbol{R}^T(\boldsymbol{\eta})\boldsymbol{K}_p\tilde{\boldsymbol{\eta}}}_{\boldsymbol{K}_d^*}\boldsymbol{\nu} - \boldsymbol{R}^T(\boldsymbol{\eta})\boldsymbol{K}_i\int_0^t \tilde{\boldsymbol{\eta}}(\tau)d\tau \tag{7.5}$$

$$\boldsymbol{K}_d^* := \boldsymbol{R}^T(\boldsymbol{\eta})\boldsymbol{K}_p\tilde{\boldsymbol{\eta}} \tag{7.6}$$

where $\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta} - \boldsymbol{\eta}_d$ is the position and attitude error. The derivative gain $\boldsymbol{K}_p$ is usually set as a diagonal matrix, resulting in $\boldsymbol{K}_d^* = \boldsymbol{K}_p$.

The integral term in the controller will compensate for the bias term in the model. The wave term is assumed to be negligible, as the controller will only be used close to berth in a harbour environment with little wave disturbances.

7.3 Assumptions

This controller assumes full state feedback, meaning that all states need to be measured or estimated. For this simulation the states are available as measurements, but in a real situation they would need to be estimated by the use of a Kalman filter or a nonlinear observer for example.

In order to implement the wind feedforward $\boldsymbol{\tau}_{wind}$ it is necessary to know the wind forces and moments as a function of wind speed and direction. These may be found by issuing scale model tests, or by using simpler models. In the simulations wind disturbances are known and used when simulating with environmental disturbances.

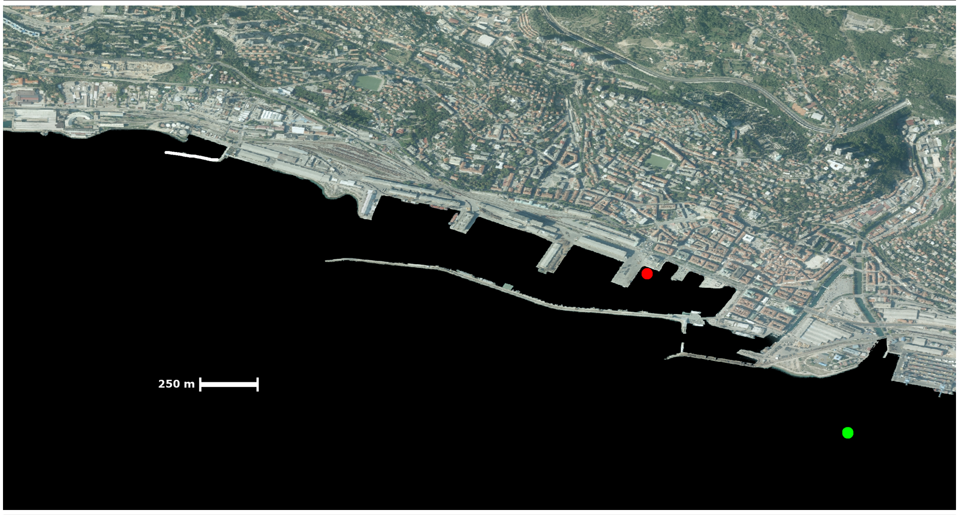
Chapter 8

Results

8.1 Path planner

The harbor of Rijeka, Croatia was chosen to be the test environment for the path planner. The harbor is showed in figure 8.1. The image has been processed to show the sea/free environment as black to simplify manipulation in MATLAB. The goal will be to plan a path from the green dot to the red dot using the A* algorithm.

Figure 8.1 The harbour of Rijeka, Croatia.

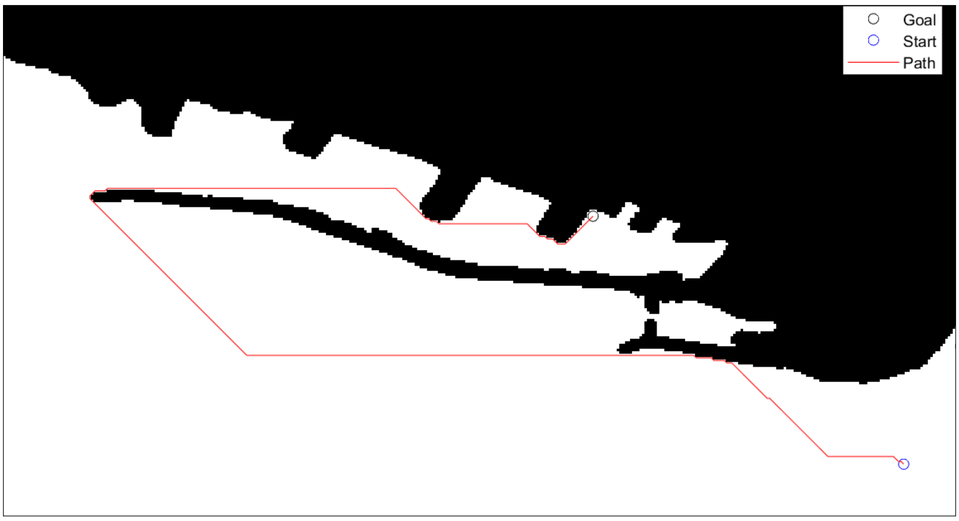


All the tests were run on this map scaled down by 80% to a size of 236 by 526 pixels. Before scaling the map was dilated by 8 pixels which corresponds to about 12 meters.

8.1.1 No augmentations

In figure 8.2 the path is generated without any augmentations to the algorithm. The path is jagged and certainly not the shortest possible. The only possible path orientations are the 8 cardinal and ordinal directions. The path is planned as close to the obstacles as possible which is fine because of the added safety distance provided by the dilation of the obstacles. The run-time of the algorithm, in this case, is 5.5 seconds.

Figure 8.2 Planned path with no augmentations.



8.1.2 Increasing connecting distance

By changing the connecting distance, clearly better paths are generated as seen in figures 8.3 and 8.4. The increased connecting distance allows for more path orientations in the grid. Increasing the connecting distance has only increased the run-time to 5.7 seconds and 7.2 seconds respectively.

Figure 8.3 Planned path with a connecting distance of 4.

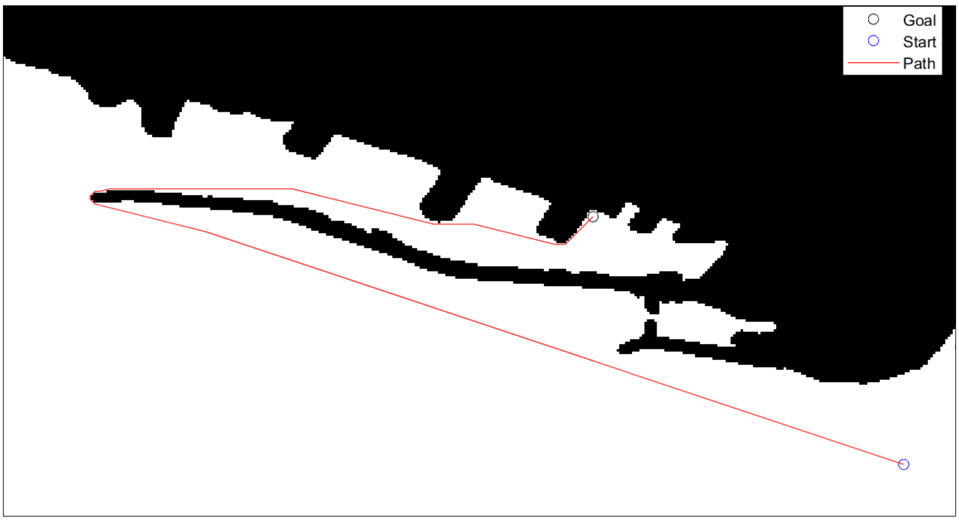
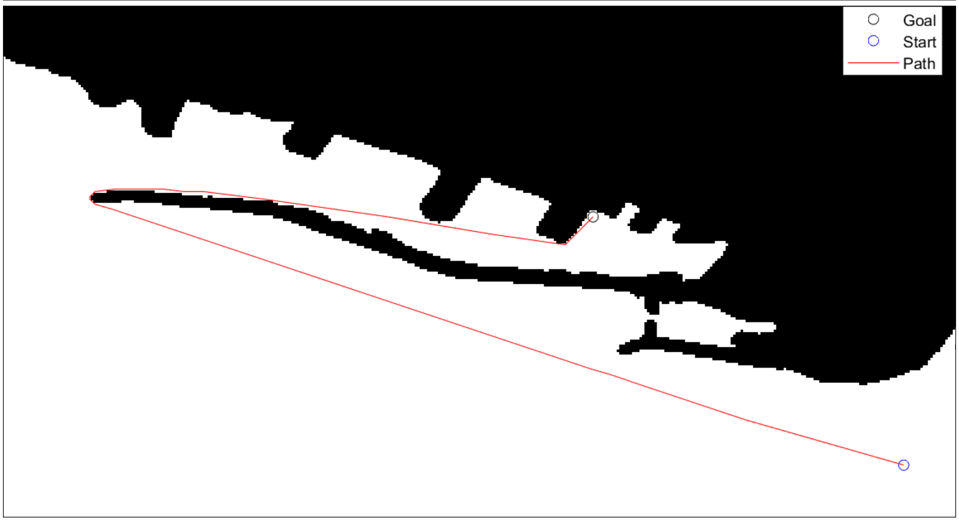


Figure 8.4 Planned path with a connecting distance of 8.



8.1.3 Penalizing closeness to obstacles

Figure 8.5 and 8.6 shows the path planned when the map has been weighted according to equations 4.1 and 4.2 respectively. The path is now a trade-off between the shortest path and the furthest from obstacles. One can see that it now curves wide from the pier instead of lying as close as possible. Slightly undesired effects are seen in figure 8.5 when the start of the path curves wide of the long pier instead of forming a straight line as in figure 8.6.

This is unnecessary behavior as the distance from the pier is already large enough. The run-time is increased to 8.4 seconds on average when the cost-map has been pre-calculated. Calculating the cost map uses an average of 13.6 seconds.

Figure 8.5 Planned path with a connecting distance of 8 and penalty for being close to obstacles according to equation 4.1.

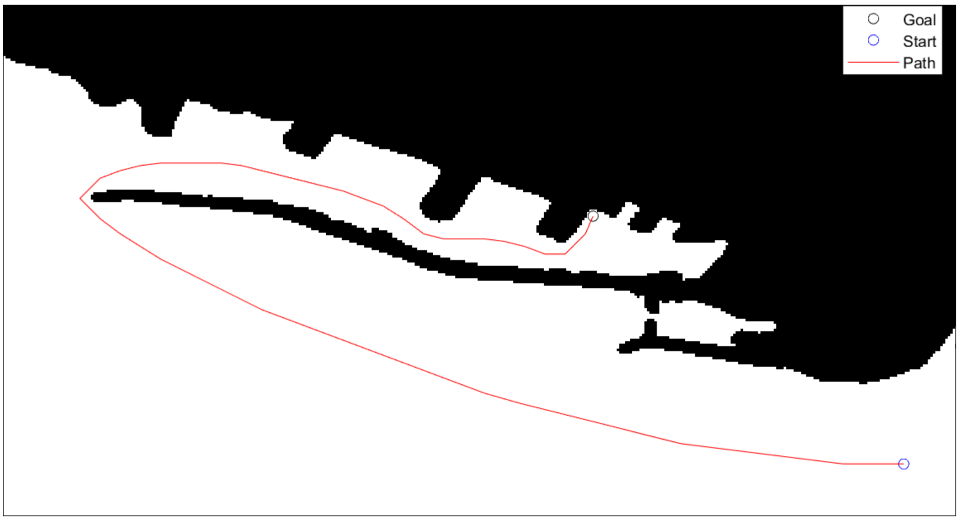
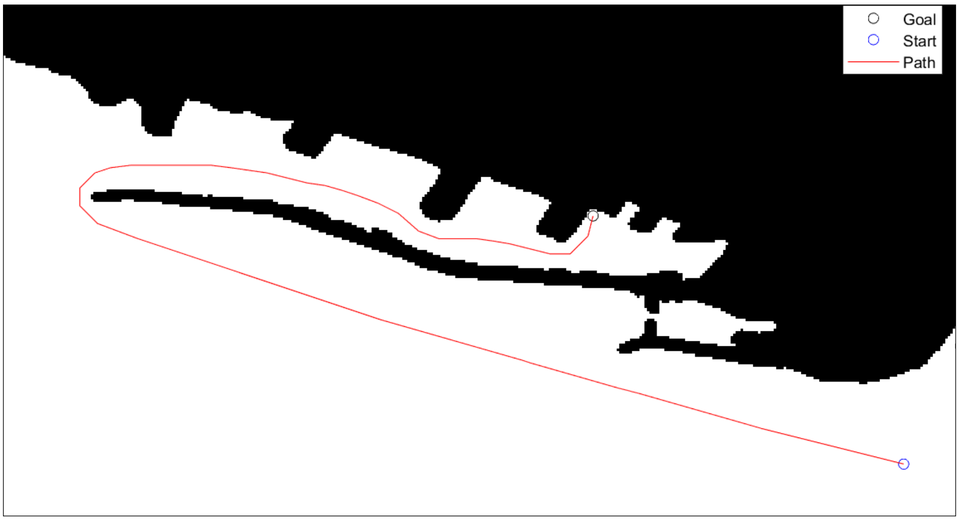


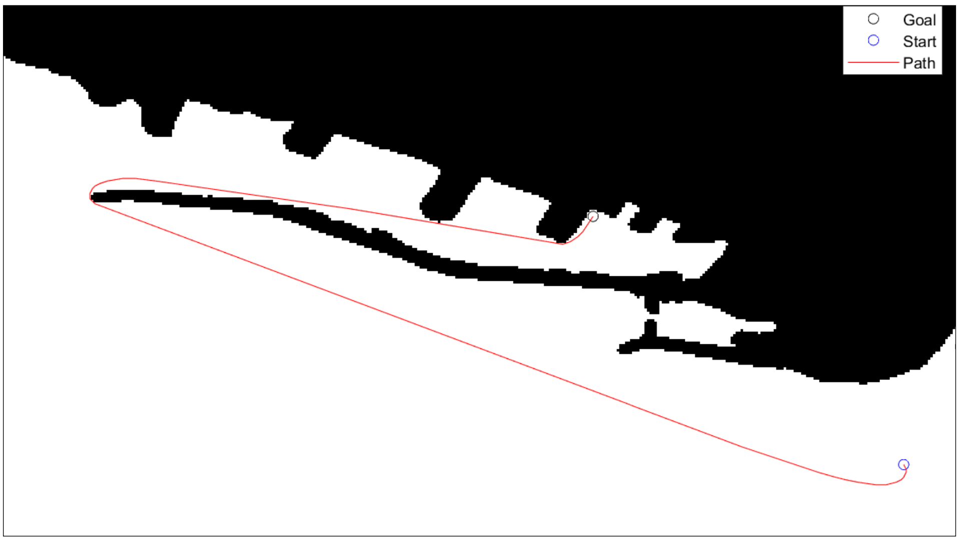
Figure 8.6 Planned path with a connecting distance of 8 and penalty for being close to obstacles according to equation 4.2.



8.1.4 Penalizing sharp turns

Figure 8.7 shows the effect of penalizing large change of path orientation as explained in subsection 4.4. There is no penalty for being close to obstacles. The algorithm is now a trade-off between the shortest path and the least curvature. The initial orientation of the vessel is towards the south-east so it has to turn around at the start of the path. In the previous tests this was not a problem as there was no penalty for immediate turns. We can see that this addition to the algorithm provides a smoother path than previously. The run-time increases to 37.2 seconds, a much larger increase than before.

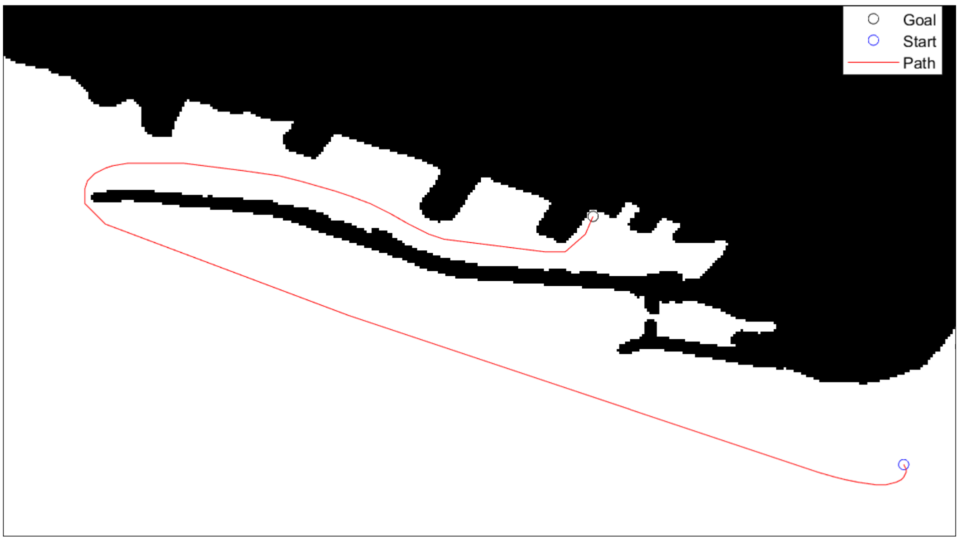
Figure 8.7 Planned path with a connecting distance of 8 and penalty for large change of path angle.



8.1.5 All augmentations

In the final test, all augmentations are used together as shown in figure 8.8. The weighting scheme from equation 4.2 is used. It is now a trade-off between the shortest path, distance from obstacles and curvature. The final path is smooth and stays well away from the obstacles. The average run-time is 42.8 seconds.

Figure 8.8 Planned path with a connecting distance of 8, penalty for being close to obstacles according to equation 4.2, and penalty for large change of path angle.



8.1.6 Discussion

The basic implementation of the A* algorithm is not suited for path planning for a ship as seen in figure 8.2. It is too jagged and will not result in the shortest path. Using all the augmentations from chapter 4 it is seen that it is possible to plan a path using the A* algorithm that is concerned with vessel dynamics while keeping well clear of stationary obstacles. Note that it will not guarantee a path which the ship can follow, but with good tuning, it may be confident of the path in most scenarios.

The weighting scheme from equation 4.2 is a clearly better choice as seen when comparing figures 8.5 and 8.6. It does not incite staying further from the obstacle than necessary and thus generates shorter paths.

The significant run-time increase penalizing path angle comes from the use of the $arctan(x)$ operator. There is room for optimization of this technique. Overall the run-time of the planner highly variable depending on the size and resolution of the map and the distance to the finish. By porting the path planner to another programming language such as C++ it will likely increase its speed significantly. However, it is not intended to be run in real time, only to update the path as it is available. On a large ship, the computation power required should be of little significance.

8.2 Path following controller

Three scenarios are presented to test the performance of the path following controller. The first scenario is the following of a straight line, with an initial position offset from the path. This scenario shows the step response of the controller, and will reveal oscillatory behaviour and convergence rate. The second scenario is the following of a circular path. This will reveal how well the controller responds to the curvature κ of the path. The final scenario is following the path generated by the path planner in section 8.1. This will show how well the path planner works, as well as the path following controller's response to a more realistic path. All the simulations are done with and without the TA system, which includes actuator saturation and realistic propulsion as discussed in section 3.2.3 and 3.2.4, for comparison.

The gains for the path following kinematic controller were found by using a grid search for pairs of K_1 and Δ. The search was preformed on the three paths presented in this section. The metric used to measure the quality of the pair was the total of the squared cross track error $\sum_i e_i^2$ where e_i is the cross track error in each time sample i. The pair which had the best score for each path was chosen.

The gains for the path following kinematic controller, as well as the yawrate and speed controller are given in table 8.1. The sample time used in the simulations is $T_s = 0.1s$, which is a compromise between simulation time and accuracy. The dynamics of a ship of this size are sufficiently slow to justify this sampling time. Experimentation with the simulator has shown that reducing the sampling time below 0.1 does not significantly impact the results. The forward speed U_d is set to $4\,\mathrm{m\,s^{-1}}$. This speed is reasonable for near coast operations.

Table 8.1: Gains and parameters used in the simulation

Δ	K_1	K_2	K_p^r	K_p^s	K_i^s	U_d	T_s
600m	0.01	0	1×10^{11}	2 000 000	0.01	4 m/s	0.1s

8.2.1 Straight line path

The straight line path is show in figure 8.9 where 8.9a shows the response without TA and 8.9b is without. The actual path taken by the ship is shown in blue, while the desired path is show in red for all figures in this section. The figures show smooth, non-oscillatory behaviour. The response with thrust allocation is noticeably slower to react. This is due to the delay caused by the rotation of the thrusters and by non instantaneous increase in power.

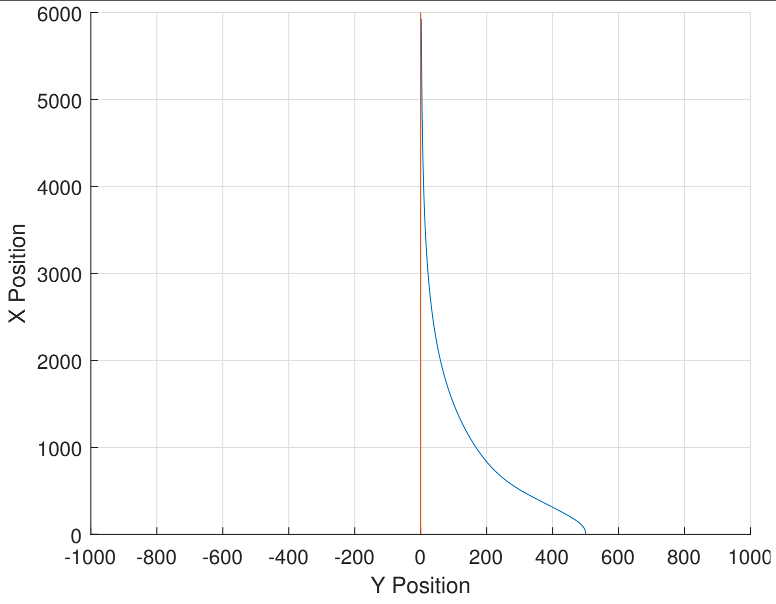
The ship does not move towards the closest point on the path before beginning to follow it. Also, the point where the ship meets the path is much farther than the lookahead distance. This is because the virtual target moves ahead on the path so as to keep the along track error s at zero. At every time instant, the controller tries to point the ship at a point located $500\,\mathrm{m}$ ahead on the path. Thus as the ship gets closer to the path the angle gets smaller.

8.2.2 Circular path

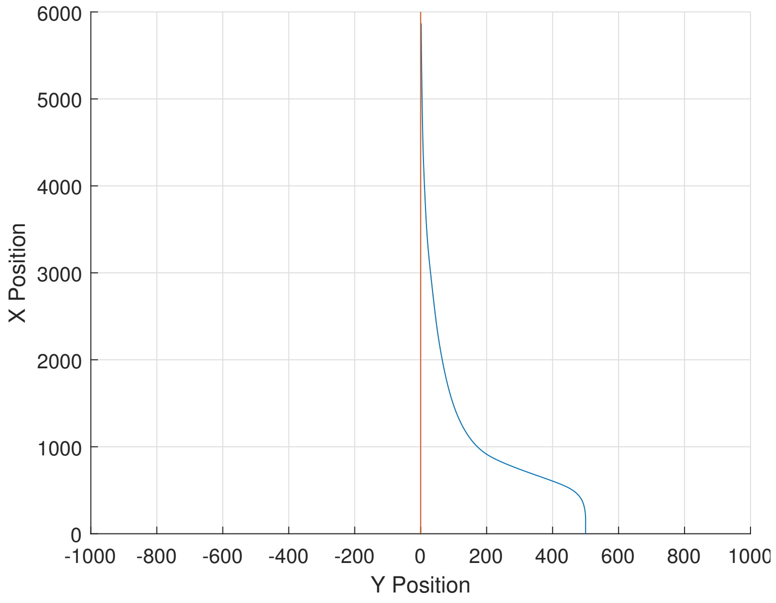
The circular path is shown in figure 8.10. This demonstrates the controllers ability to follow curved paths is smooth and non-oscillatory. Again the thrust allocation is much slower, as seen in the previous section. Both the figures show a small steady state error.

The radius of the circle was chosen to be slightly larger than the maximum turn rate of the ship model travelling at $4\,\mathrm{m\,s^{-1}}$. This radius was of found by setting the TA's desired

Figure 8.9 Test of the path following controller with an offset initial position for a straight line path



(a) Without TA



(b) With TA

yaw moment to infinity while controlling speed to $4\,\mathrm{m\,s^{-1}}$. The resulting circular path had a radius of $1500\,\mathrm{m}$.

8.2.3 Path planner path

The dimensions of the ship used in these simulations are much too large to dock at the harbour environment presented in section 8.1. The path used in this section is a scaled up version of the one found in section 8.1 by a factor of 10. The results can be seen in figure 8.11. One can see that the path path is almost perfectly followed for the ideal case without TA.

Figure 8.11b again shows the delay in actuation caused by the TA system. To the left in the figure where the ship deviates from the path one can see that the ship is unable to follow such a sharp bend at cruise speed. At $Y = -2000\,\mathrm{m}$ there is also a deviation from the path. This is likely caused by the filtering of the path curvature, which causes a time delay in the signal. Thus the filtered curvature is nonzero on straight path segments close to curves. This will cause the controller to keep turning due to curvature, even as the path straightens out.

8.2.4 Discussion

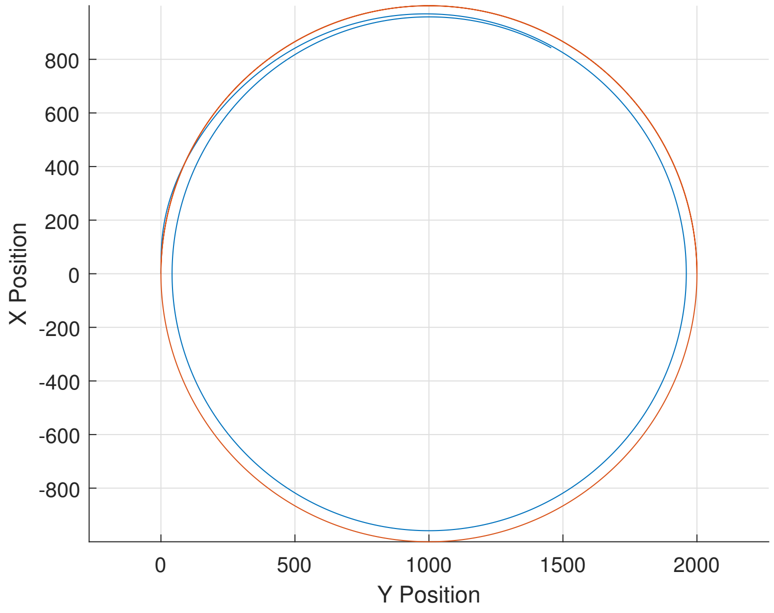
The path following kinematic controller performs satisfactory for path following. The results here show that it is limited by the nature of the path, and the response of the TA system. Subsection 8.2.3 demonstrates the path planner's ability to create a feasible path. Although the path has been scaled up to fit the restrictions of the ship model, it can easily be tuned to create paths of this scale.

While following the circular path a steady state offset is observed. This may be caused by the curvature part equation 5.2, κU_d, having no gain. The controller is based on a linearized model of the ships dynamics, and the constant $\left(1 - \frac{(m - X_{\dot{u}})}{(m - Y_{\dot{v}})}\right)^{-1}$ may not sufficiently represent the real system. It may be possible to reduce steady state error when following constant curves by designing a gain parameter for κU_d.

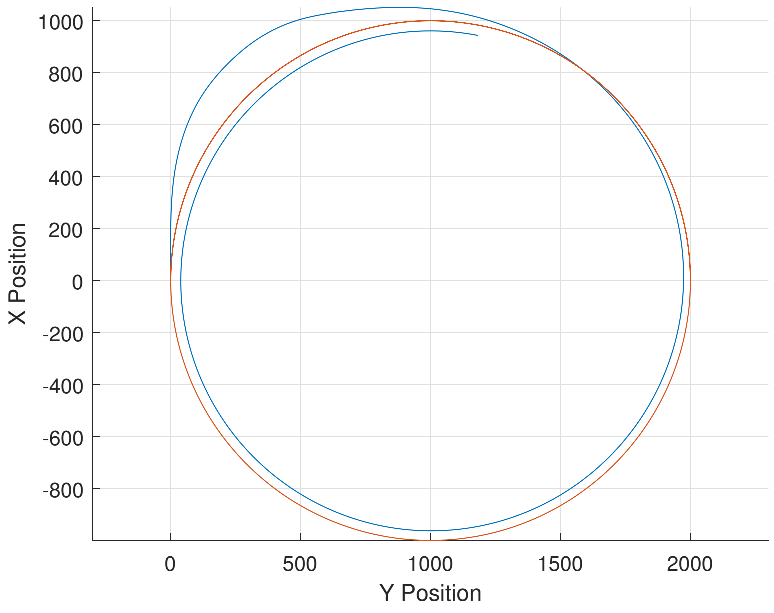
The design of keeping the along track error s constant allows for the addition of an independent speed controller. Such a controller may look ahead on the path and regulate the speed according to the curvature of the upcoming path.

A possible design modification would be to keep the virtual target stationary for large cross track errors e. This would result in a steeper approach to the path. However, this might cause unpredictable behaviour.

Figure 8.10 Test of the path following controller for a circular path

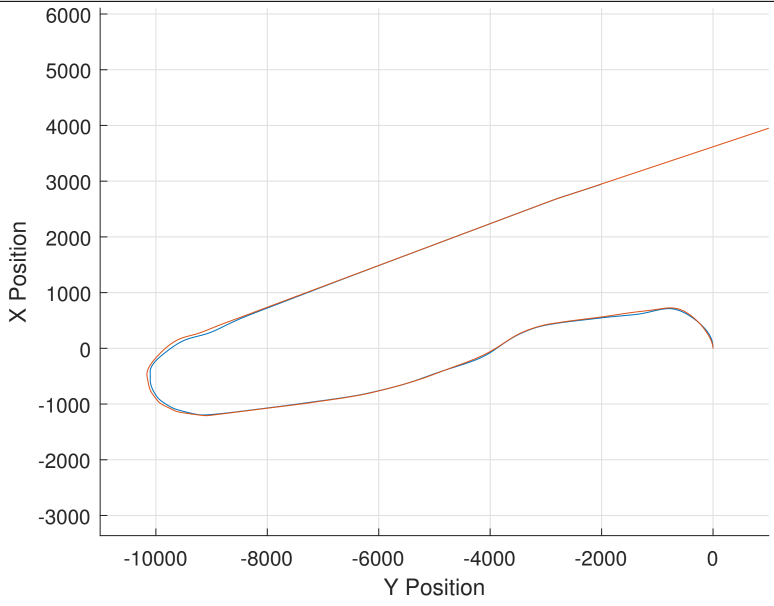


(a) Without TA

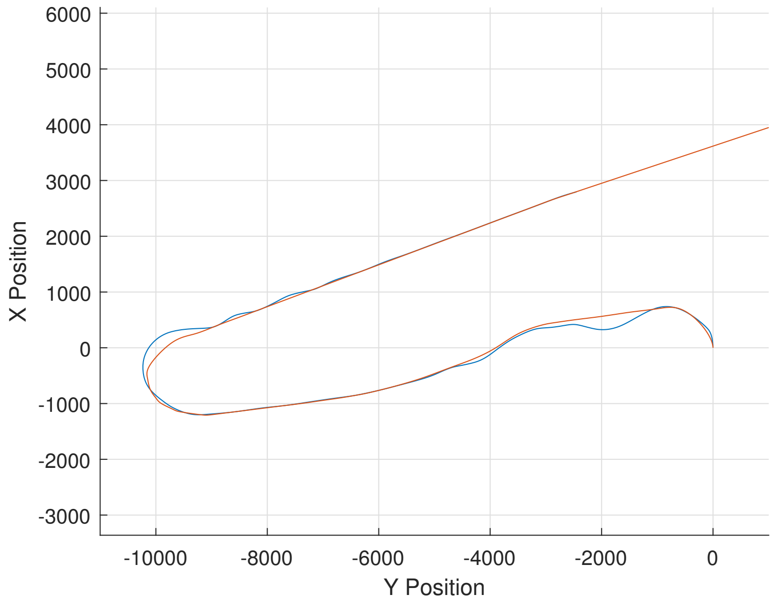


(b) With TA

Figure 8.11 Test of the path following controller for a the planned path



(a) Without TA



(b) With TA

8.3 Dynamic collision avoidance

Four different test scenarios were run too test the performance of the VPM method as a dynamic obstacle avoidance method. The first scenario is avoiding a head on collision with a single circular obstacle. This will demonstrate the basic ability of the method, as well as the functionality of the rotor function. The second scenario will be for multiple clustered obstacles. The third scenario is following a narrow passageway. The final scenario will be to test the path provided by the path planner with some obstacles added. All the simulations are done with and without the TA system.

The gains A_o, $A_{\omega 0}$ and A_r for the obstacles and waypoint are given in table 8.2. Also included in the table is the agent sensor range Δ_a and the agent Field Of View (FOV).

Table 8.2: Gains and parameters used in the simulation

A_o	A_r	$A_{\omega 0}$	Δ_a	FOV
1000	2000	0.3	$1500\,\mathrm{m}$	±45°

8.3.1 Single obstacle

By approaching the obstacle head on without the rotor function, the VPM method would result in a local minima. This is because the repulsive force is directed opposite of the attractive force and at some point they will cancel each other out. The rotor function is demonstrated in figure 8.12. One can see that the VPM method successfully avoids collision for single obstacles. The deviation margin of deviation is quite large at around $500\,\mathrm{m}$ for the non-TA case, and $800\,\mathrm{m}$ with TA.

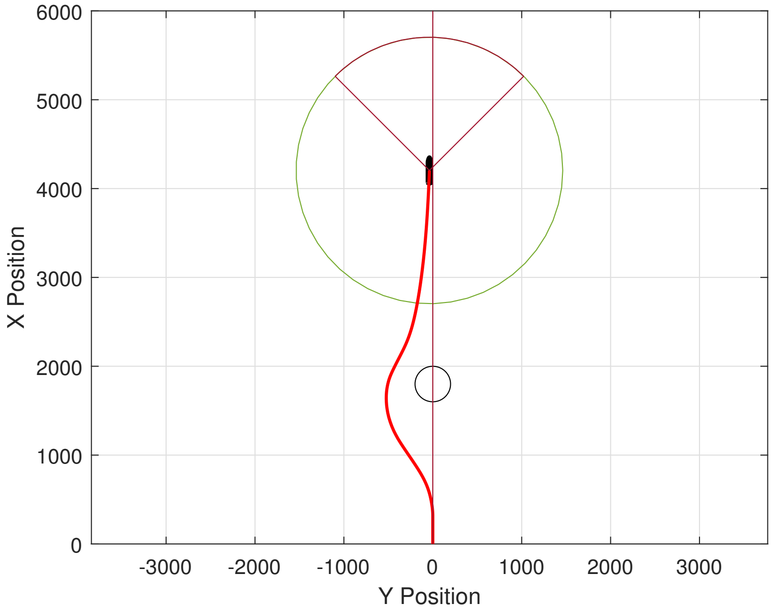
8.3.2 Multiple clustered obstacles

In the case of multiple clustered obstacles the method is unable to discern if it can safely pass between them. Figure 8.13 depicts such a scenario. Here the ship tries to pass between the obstacles, as their rotor functions command a force the shortest way around each individual obstacle. In figure 8.13a one can see that at first the ship attempts to pass on right of the cluster. As it gets close the rightmost obstacle's rotation field forces the ship towards the left again. The result is collision. This scenario demonstrates the limitations of the rotor function to eliminate local minima.

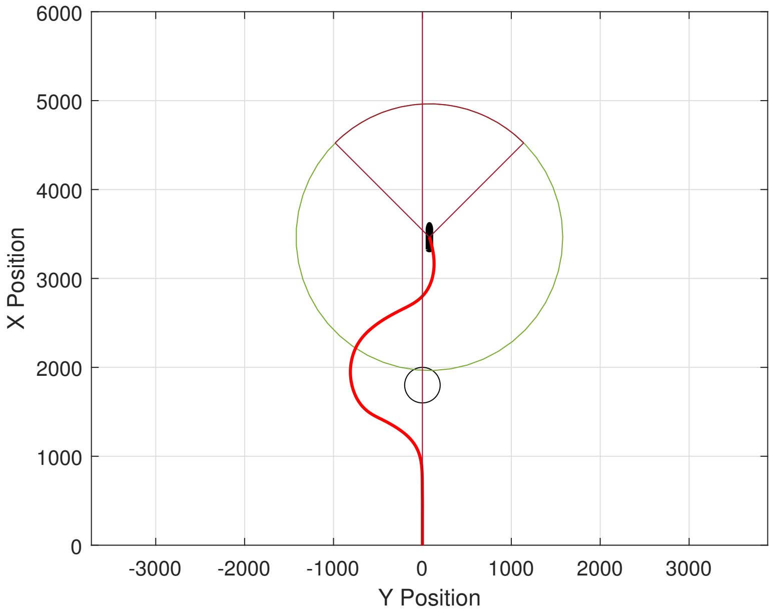
8.3.3 Narrow passage

This scenario demonstrates how the method can navigate multiple obstacles if they are sufficiently spaced, such that a feasible path can be followed between them. The results are show in in figure 8.14. The repulsive forces of each obstacle cancel out in the middle, resulting in the ship being commanded by the attractive force of the waypoint, as well as the rotor function directed parallel to the obstacles. The behaviour with TA is oscillatory because of the delay in actuation.

Figure 8.12 Test of the VPM controller for a single obstacle

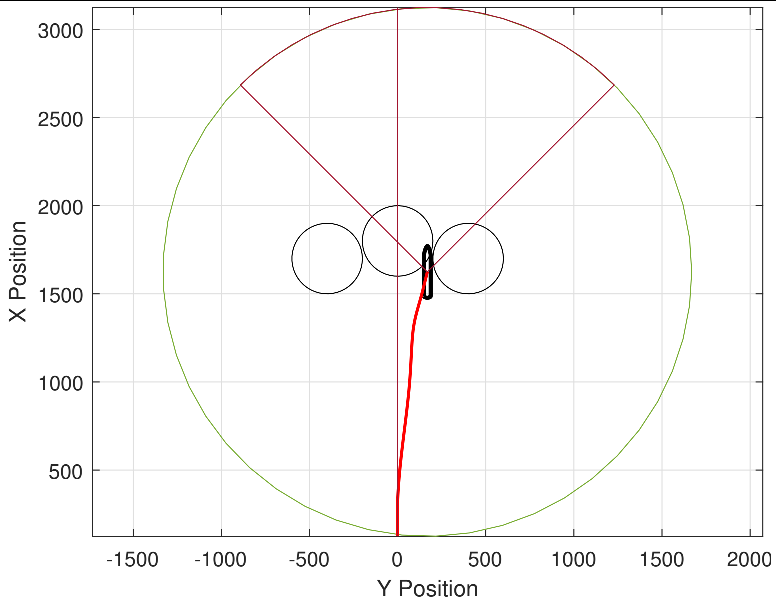


(a) Without TA

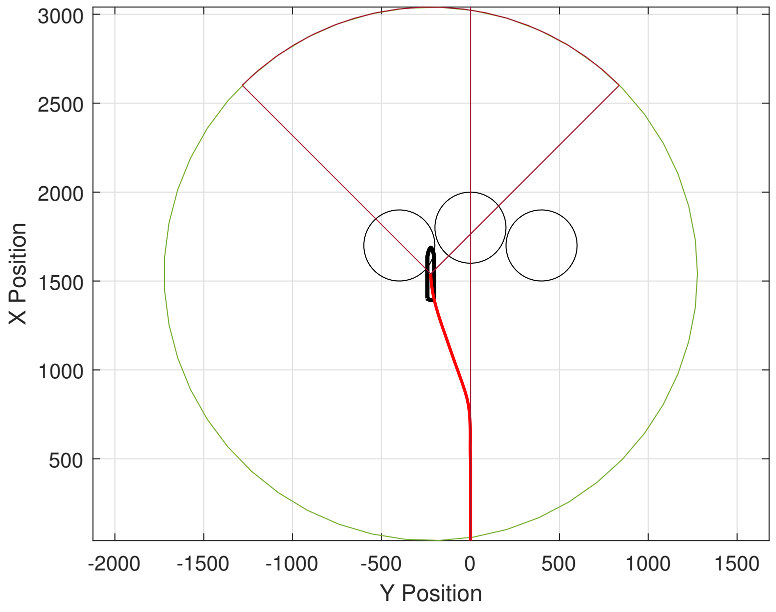


(b) With TA

Figure 8.13 Test of the VPM controller for a multiple clustered obstacles

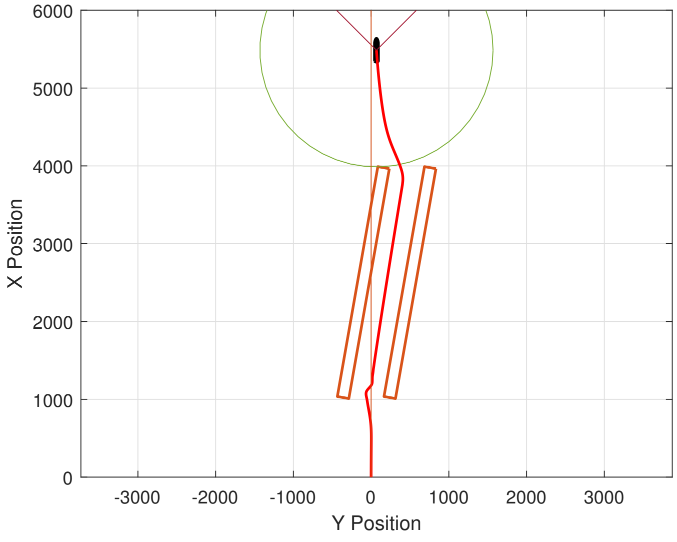


(a) Without TA

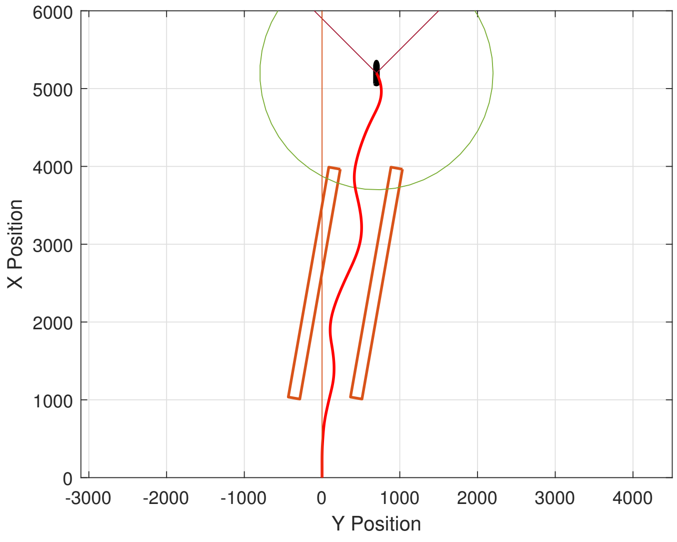


(b) With TA

Figure 8.14 Test of the VPM controller for a narrow passage



(a) Without TA



(b) With TA

8.3.4 Planned path

This scenario is a constructed example of unknown obstacles appearing in the planned path from the path planner. It is clear from figure 8.15 that the method succeeds in following the path while avoiding these obstacles. As noted in chapter 6 the waypoint is set as the point along the path where the along track error s equals the sensor range, when an obstacle is detected. The effect of this can be seen in the clockwise turn to the left in the figure. Here the controller sees the big rectangle and places it's waypoint on the northern part of the path after the curve. Therefore the controller ignores the planned path and cuts the corner.

8.3.5 Discussion

The VPM method for dynamic obstacle avoidance seems to work in most of these constructed examples, however, it suffers from unreliability in clustered waterspaces.

Combining clustered obstacles

In order to avoid the problem of clustered obstacles demonstrated in figure 8.13, a method of combining multiple obstacles into a single large one could be implemented. This method would have to decide if passing between two obstacles is at all possible, and if not combine them into a single one. This would remove the issue of having two rotor fields in opposite directions forcing the ship into a collision. As demonstrated in the other scenarios, the VPM method can handle navigating in environments where there are not tightly clustered obstacles.

Potential functions

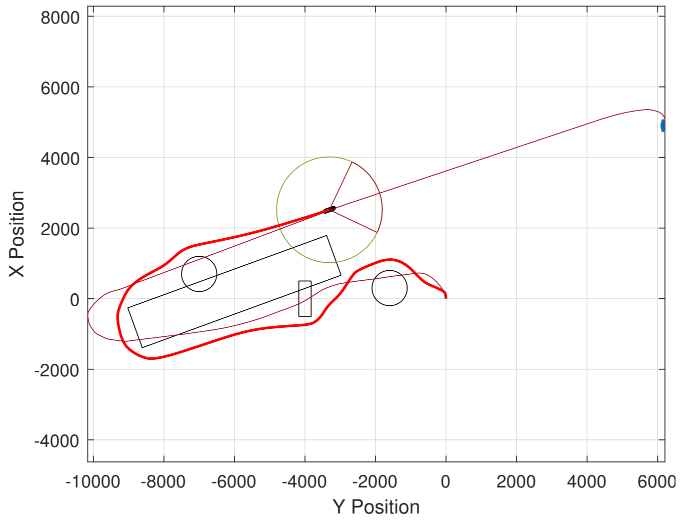
The current implementation of the VPM method as a means for dynamic collision avoidance only issues a yawrate command based on the instantaneous location of the ship relative to the visible obstacles. The potential functions used for this implementation may be unsuited for this usage. The decentralized control function from each obstacle $-\nabla P_o(\boldsymbol{x})$ is negligible compared to the decentralized control function the waypoint $-\nabla P_\omega(\boldsymbol{x})$ for large distances. The opposite is true for small distances. This can be seen in figure 6.1a versus 6.1b. The total distance they are of comparable magnitude is incredibly short. The gains A_o and $A_{\omega 0}$ only decide at what the critical distance from the obstacle they at a comparable magnitude. The result of this is that the controller commands the ship to travel straight towards the waypoint, and then as it reaches the critical distance it commands the ship straight away from the obstacle.

Changing the potential functions to overlap for a larger total distance will allow the ship more time to react to the approaching obstacle, instead of meeting a "wall" when it gets a certain distance from it.

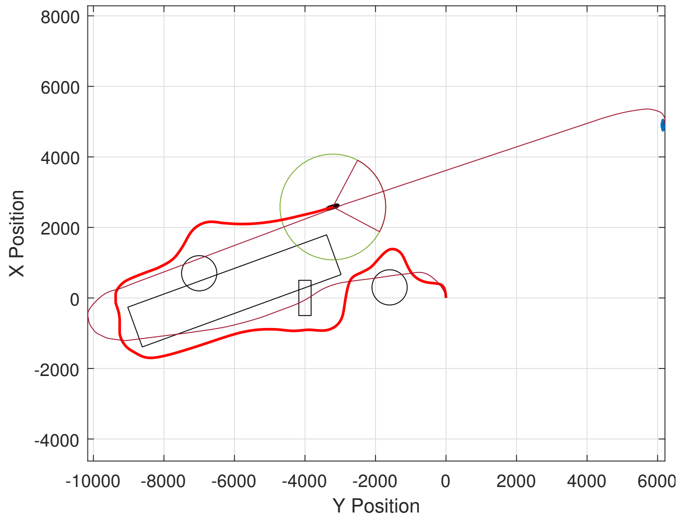
MPC

Another possible improvement to the VPM would be to implement a MPC controller. These are discussed in section 3.4.3. Such a controller would look ahead and optimize its desired yawrate to avoid hitting the "wall" where the repulsive and rotor potentials dominate as the

Figure 8.15 Test of the VPM controller for the planned path



(a) Without TA



(b) With TA

distance to the obstacle gets smaller. A MPC might work well with the current potential functions.

Moving obstacles

Moving obstacles are not considered in this thesis. The VPM method is likely capable of handling slowly moving obstacles by just changing their observed location at each time step. Additional methods would be required to handle the following of COLREGS intelligently. One such method could be to adjust the rotor field of an obstacle if it is classified as a ship. The rotor field would then be directed so as to force the ship to pass the obstacle ship in accordance of the COLREGS.

As discussed in chapter 2, Larson et al. (2006) suggests using the POA method to handle the uncertainty of the future location of moving obstacles. The VPM method is fully capable of avoiding a POA. Therefore only the POA method needs to be implemented in order to make the VPM method capable of avoiding dynamic obstacles.

8.4 Berthing

The scenario chosen to test the DP berthing controller is a simple step of $100\,\mathrm{m}$ in positive sway direction and 10° clockwise. This will demonstrate the planned usage scenario for the controller. The simulation has been done without environmental disturbances as these are assumed negligible in a calm harbour environment. The simulations were conducted with and without TA.

The gains for the DP controller are given in tables 8.3 and 8.4 for the tests without TA and with TA respectively. To avoid integrator windup in the DP controller a control signal must be sent to disable the integrator when the actuators reach saturation. Due to the black box nature of the TA system, such a control signal was unavailable. Therefore the integrator gains are set to zero for the the tests with TA. The gains have been found through trial and error and have been found satisfactory for these tests. The reference model parameters were chosen for a critically damped response and a sufficiently slow change of position. They are given in table 8.5.

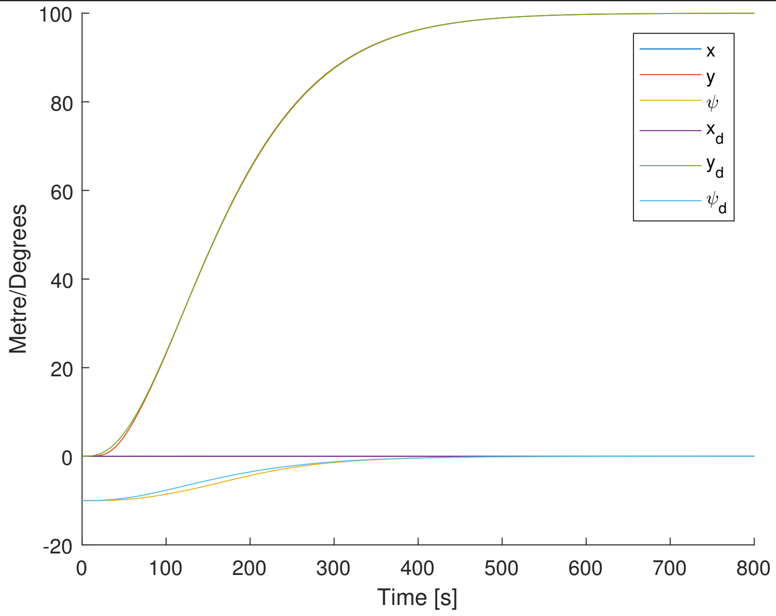
Table 8.3: DP gains without TA

Kp	diag($\begin{bmatrix}2\times10^6 & 2\times10^6 & 0 & 0 & 0 & 2\times10^9\end{bmatrix}$
Kd	diag($\begin{bmatrix}3\times10^7 & 3\times10^7 & 0 & 0 & 0 & 3\times10^7\end{bmatrix}$
Ki	diag($\begin{bmatrix}3\times10^5 & 3\times10^5 & 0 & 0 & 0 & 5\times10^4\end{bmatrix}$

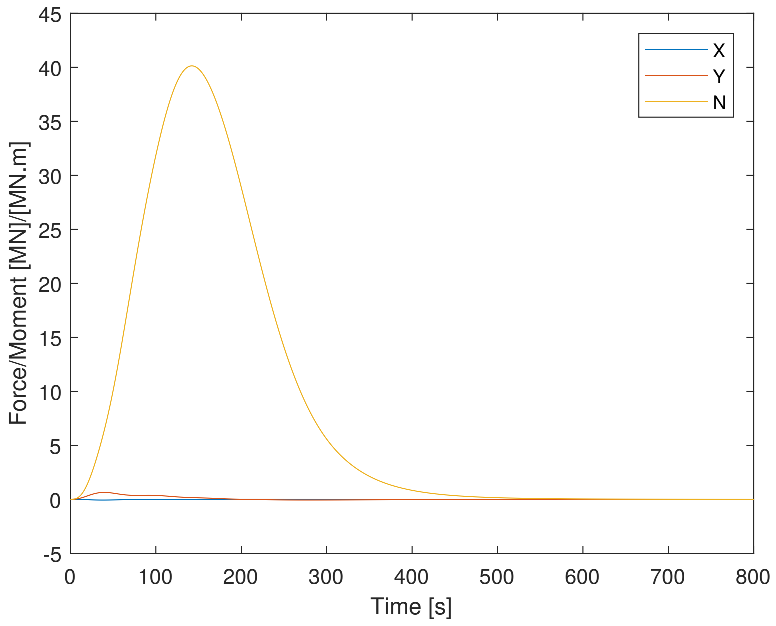
8.4.1 Without TA

The results are shown in figure 8.16. The results show that the controller can quickly follow the reference trajectory, η_d, generated by the reference model. The gains have been tuned to this reference model and without TA in mind.

Figure 8.16 Test of the DP controller for a step in reference of $100\,\mathrm{m}$ and $10°$. Without TA



(a) η and η_d



(b) Forces and moment

Table 8.4: DP gains with TA

Kp	$\mathrm{diag}(\begin{bmatrix}2\times10^5 & 2\times10^5 & 0 & 0 & 0 & 6\times10^7\end{bmatrix}$
Kd	$\mathrm{diag}(\begin{bmatrix}6\times10^6 & 6\times10^6 & 0 & 0 & 0 & 6\times10^5\end{bmatrix}$
Ki	$\mathbf{0}$

Table 8.5: Reference model parameters

ω_n	ζ
$\frac{1}{60}$	1

8.4.2 With TA

With TA included the results are noticeably slower. They can be seen in figures 8.17 and 8.18. In figure 8.17a one can see that the heading deviates significantly from the reference. The explanation can be found in figure 8.18b. The actual yaw moment τ_{real} does not follow the desired moment $\tau_{desired}$. This is likely because the bow tunnel thruster activates in order to follow the desired sway force, this in turn causes a large yaw moment. Time delay from the rotation of the azimuth thrusters means that they are unable to compensate in time.

A second test without a step in heading reference is shown in figure 8.19. This test shows that the heading is impacted by a change in position reference. This is an undesirable effect, as the bow or stern of the ship may collide as it approaches the quay.

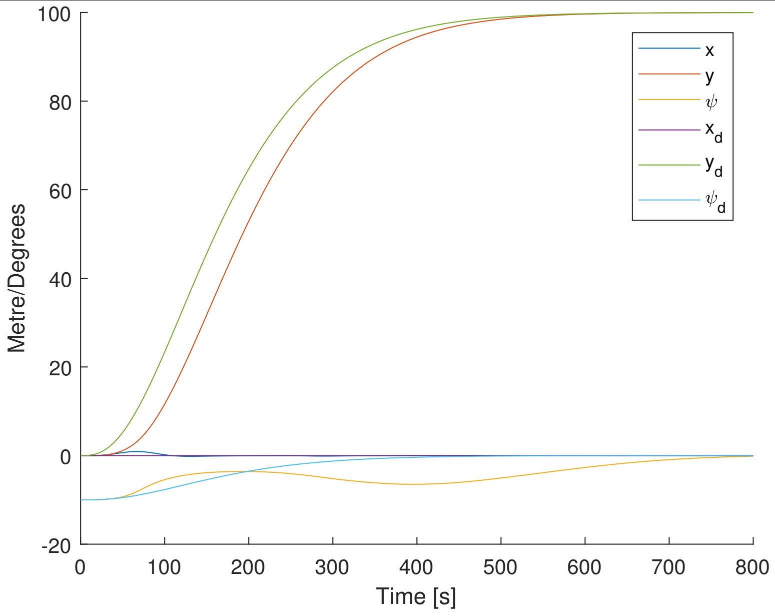
The gains have needed to be tuned with TA in mind, as the gains given in table 8.3 cause the controller to be unstable.

8.4.3 Discussion

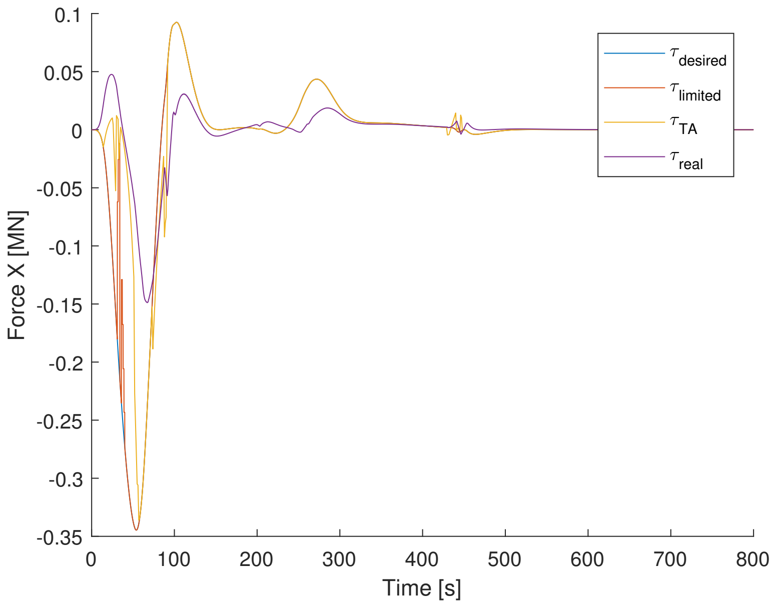
The results show almost satisfactory performance from the DP controller, both with and without TA. Figure 8.19 shows the problem with the controller. The heading angle is significantly impacted by a step in position reference, which is unacceptable when approaching the quay. The length of the ship is $294\,\mathrm{m}$, a small heading deviation of $5°$ will result in the bow or stern to be $13\,\mathrm{m}$ closer to the quay than desired. In order to increase the performance of the controller, better tuning of the gains must be done.

The controller has not been tested with environmental disturbances. The assumption of small disturbances in the harbour environment may be incorrect in many ports. Especially the current from the tide, and strong winds may still be present in a harbour environment. The controller must be tested and perform satisfactory in the case of these disturbances in order to be production ready.

Figure 8.17 Test of the DP controller for a step in reference of $100\,\mathrm{m}$ and $10°$. With TA.

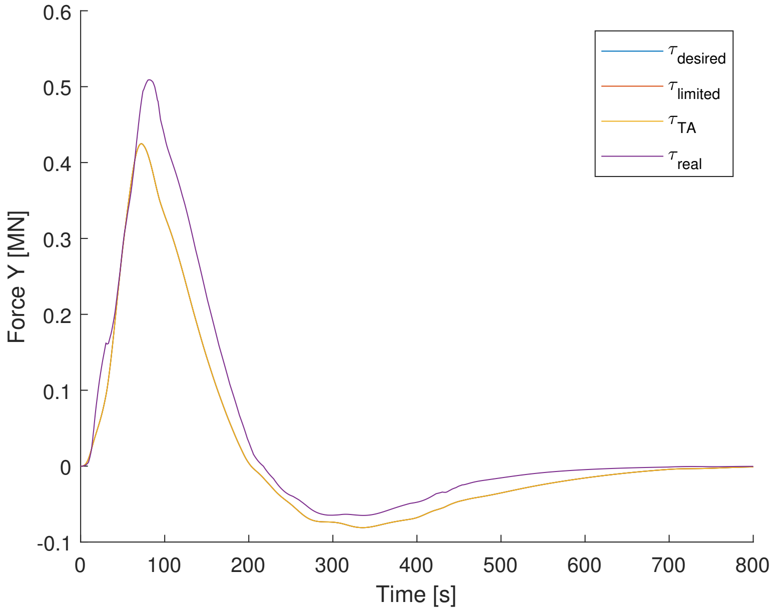


(a) η and η_d

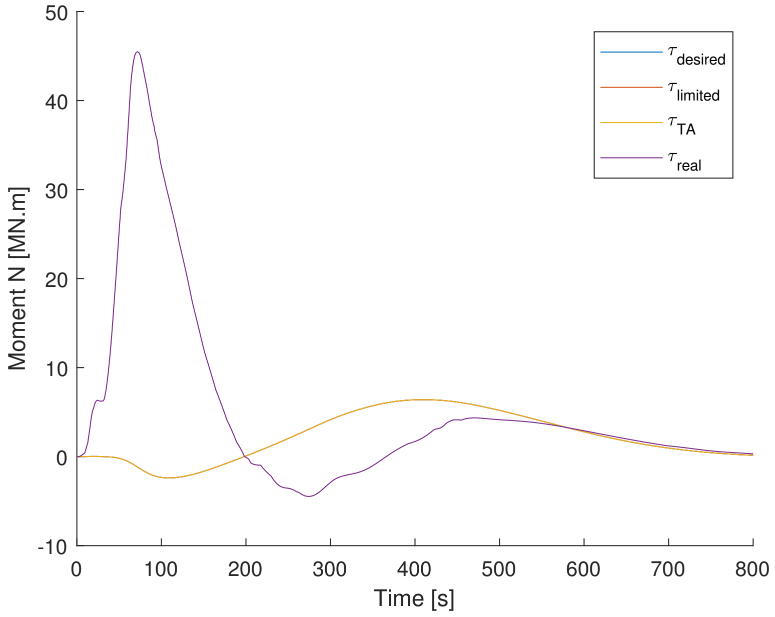


(b) Force in surge

Figure 8.18 Test of the DP controller for a step in reference of $100\,\mathrm{m}$ and $10°$. With TA.

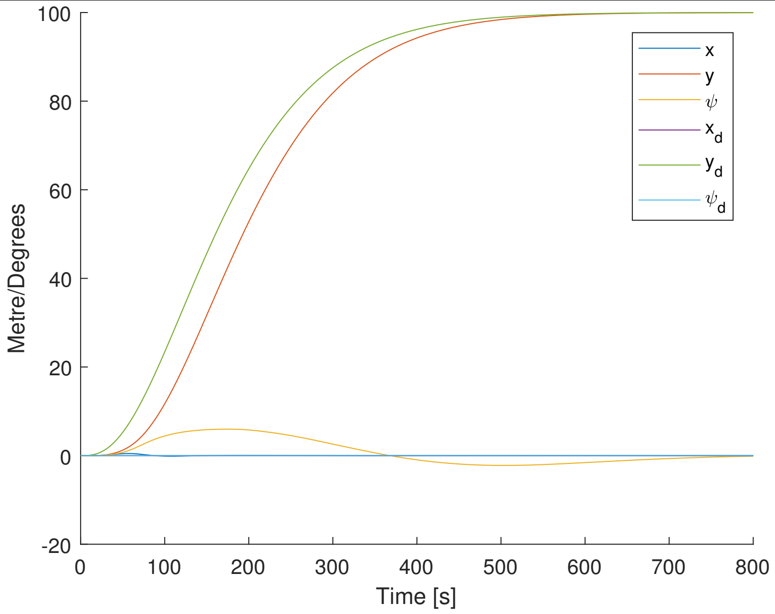


(a) Force in sway



(b) Moment in yaw

Figure 8.19 Test of the DP controller for a step in reference of $100\,\mathrm{m}$ and $0°$. With TA.



8.5 Overall system

When combining all the results of this chapter it seems reasonable to say that the problem defined in section 1.2 can be solved with a few improvements to the methods. The system is capable of planning a path from its current position to the berthing spot. It is capable of following this path while avoiding unforeseen static obstacles. With the addition of POA method it should be capable of avoiding dynamic obstacles. By better tuning of the current DP controller, or by using an industry standard DP controller it should be able to steer the final approach to berth.

Chapter 9

Conclusion and future work

The aim of this thesis was to create a complete autonomous berthing system for a short sea shipping vessel. In order to achieve this, four different systems have been implemented. First the path planner determines the shortest path to the berthing spot. Then the path follower controls the ship to follow this path. The dynamic collision avoidance system detects and avoids any unforeseen obstacle in the way. Finally the DP controller carefully steers the ship the final distance to the quay. These systems need to communicate with each other and work in conjunction with each other. The results of this thesis are solely focused on each system in isolation.

The simulation results show that the performance of the path planner, follower and the dynamic collision avoidance have been satisfactory. The DP controller for the final approach to berth has been shown to be unsatisfactory with the current tuning. The system has not been made COLREGS compliant, nor tested with environmental disturbances to show robustness.

9.1 Future work

What remains to be done for the autonomous berthing system is to make it production ready and to test it on an industrial simulator. Simulations with environmental disturbances present need to be run. To achieve this there are many improvements to be made to the system.

Starting with the path planner, it needs to be adapted to use industry mapping software. Currently it only uses a hand processed bitmap image of the environment. The path planner should communicate with the obstacle detection system, and take advantage of new information gained of the environment. This will allow it to re-plan the path should the ship find itself in a local minima.

A more intelligent speed controller would increase the performance of both the path following controller and the dynamic obstacle avoidance system. Such a controller would take into account upcoming path curvature, or obstacles and set the desired speed accordingly. A system for emergency braking should also be included in the speed controller.

The dynamic collision avoidance system is currently not robust. New potential functions for obstacle repulsion and rotor fields should be explored. The new functions should take account for the slow dynamics of the ship. Using a MPC should also be explored. A MPC can add robustness to the system because it predicts the ships trajectory and acts accordingly. This can be combined with the speed controller to give better desired speeds, or an emergency braking signal.

The POA method or a similar method should be implemented so that the dynamic collision avoidance system can account for dynamic obstacles.

A method for combining clustered obstacles into single larger ones needs to be implemented to ensure the robustness of the VPM.

The DP controller for berthing should be tuned for better response, or exchanged for a better industry standard controller.

Bibliography

Barisic, M. (2012). *Guidance Of Formations Of Autonomous Underwater Vehicles By Virtual Potential Method*, PhD thesis, University of Zagreb.

Barisic, M., Vukic, Z. and Miskovic, N. (2007a). Kinematic simulative analysis of virtual potential field method for auv trajectory planning, *2007 Mediterranean Conference on Control Automation*, pp. 1–6.

Barisic, M., Vukic, Z. and Miskovic, N. (2007b). A kinematic virtual potentials trajectory planner for auv-s, *IFAC Proceedings Volumes*, Vol. 40.

Barisic, M., Vukic, Z., Miskovic, N. and Tovornik, B. (2007). Aub formations achieved by virtual potentials trajectory planning in a simulated environment, *IFAC Proceedings Volumes*, Vol. 40, pp. 135 – 140. 7th IFAC Conference on Control Applications in Marine Systems.
URL: *http://www.sciencedirect.com/science/article/pii/S147466701532084X*

BBC (2014). Uk to allow driverless cars on public roads in january. [Online; accessed 10-December-2017].
URL: *"http://www.bbc.com/news/technology-28551069"*

Bu, R., Liu, Z. and Li, T. (2007). Nonlinear sliding mode berthing control of underactuated surface ships, *2007 IEEE International Conference on Control and Automation*, pp. 1371–1376.
URL: *http://ieeexplore.ieee.org/document/4376584/*

Djouani, K. and Hamam, Y. (1994). Ship optimal path planning and artifical neural nets for berthing, *OCEANS '94. 'Oceans Engineering for Today's Technology and Tomorrow's Preservation.' Proceedings*, Vol. 1, pp. I/785–I/790 vol.1.
URL: *http://ieeexplore.ieee.org/document/363957/*

Djouani, K. and Hamam, Y. (1995). Minimum time-energy trajectory planning for automatic ship berthing, *IEEE Journal of Oceanic Engineering* **20**(1): 4–12.
URL: *http://ieeexplore.ieee.org/document/380251/*

E.U.CORDIS Research Program CitynetMobil. (2013). Driverless cars take to the road. [Online; accessed 10-December-2017].
URL: `"http://cordis.europa.eu/result/rcn/90263_en.html"`

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley & Sons, Ltd.

Kørte, S. Ø. (2011). *Guidance and Control Strategies for UUVs*, Master's thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics., Norway.

Larson, J., Bruch, M. and Ebken, J. (2006). Autonomous navigation and obstacle avoidance for unmanned surface vehicles, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Vol. 6230.

Loe, Ø. (2007). Collision avoidance concepts for marine surface craft., *Technical report*, Norwegian University of Science and Technology, Department of Engineering Cybernetics., Norway.

Loe, Ø. (2008). *Collision Avoidance for Unmanned Surface Vehicles*, Master's thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics., Norway.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*, Springer.

Okazaki, T. and Ohtsu, K. (2008). A study on ship berthing support system - minimum time berthing control, *2008 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1522–1527.
URL: `http://ieeexplore.ieee.org/document/4811502/`

SAE International (2016). U.S. Department of Transportation's New Policy on Automated Vehicles Adopts SAE International's Levels of Automation for Defining Driving Automation in On-Road Motor Vehicles. [Online; accessed 10-December-2017].
URL: `"https://www.sae.org/news/3544/"`

Skjetne, R., Fossen, T. I. and Kokotović, P. V. (2005). Adaptive maneuvering, with experiments, for a model ship in a marine control laboratory, *Automatica* **41**(2): 289–298.
URL: `http://www.sciencedirect.com/science/article/pii/S0005109804003024`

Ueland, E. S., Skjetne, R. and Dahl, A. R. (2017). Marine autonomous exploration using a lidar and slam, *ASME 2017 36th International Conference on Ocean, Offshore and Arctic Engineering*, Vol. 6.
URL: `http://dx.doi.org/10.1115/OMAE2017-61880`

Wikipedia (2017). A* search algorithm — wikipedia, the free encyclopedia. [Online; accessed 24-November-2017].
URL: `"https://en.wikipedia.org/w/index.php?title=A*_search_algorithm&oldid=811827433"`

Yao, Z., Hearn, G. E. and Sen, P. (1997). A multivariable neural controller for automatic ship berthing, *IEEE Control Systems* **17**(4): 31–45.
URL: `http://ieeexplore.ieee.org/document/608535/`