NTNU
Norwegian University of
Science and Technology

COLREGS Compatible Motion Planning for Autonomous Surface Vessels

Maren Kristine Eidal

Marine Technology
Submission date: June 2018
Supervisor: Vahid Hassani, IMT

Norwegian University of Science and Technology
Department of Marine Technology

Preface

This thesis has been written throughout the spring of 2018, as a final chapter to obtain a Master of Science degree from the Department of Marine Technology at the Norwegian University of Science and Technology (NTNU) in Trondheim.

The focal point of the research has been motion planning for autonomous surface vessels; an intricate topic that can be solved by a wide variety of approaches. I would therefore like to thank my supervisor, Vahid Hassani, for allowing me to follow through on my own ideas and guiding me in the right direction when necessary. Further thanks are directed to Amir R. Nejad, my co-supervisor, who helped me regain focus at times I found my thesis difficult to master.

Finally, a special thanks to Kaja Bremer, for being a valuable discussion partner and for making me laugh throughout my five years at NTNU.

Trondheim, 2018-06-11

Maren Kristine Eidal

Summary

In the past decades, autonomy has gone from a technology of the future to an inevitability. Driver-less cars are already on the roads, and autonomous surface vessels (ASVs) are well within range. With this increase in autonomy, follows a demand for robust path planning and collision avoidance methods. Further, ASVs have to share their working environment with other manned vessels. This calls for an additional element of the motion planning problem, in that the ASVs should follow the traffic rules outlined by the International Regulations for Preventing Collisions at Sea (COLREGS).

This thesis investigates how a stochastic approach, in the form of the Path-of-Probability (POP) algorithm, can be used within a motion planner to avoid multiple static and dynamic obstacles. Moreover, a feasibility design by the means of a simulator is completed, in which the simulator consists of a vessel model that is fitted with a guidance and control system. With these systems in place, a range of scenarios are used to assess the motion planner.

It was found that merging the POP algorithm with an A* search was needed to obtain decisive results, as the POP algorithm struggled in the encounter with multiple, static obstacles. For collision avoidance, the motion planning algorithm performed better. Much of this improvement is credited to the addition of virtual target points, which ensured that the ASV adhered to COLREGS in three separate collision scenarios.

Even though the POP algorithm endures flaws, its stochastic approach is still considered as one of its strengths. This, as the working environment for all ASVs is stochastic by nature. It is therefore vital to incorporate these uncertainties into a motion planner.

Sammendrag

I de siste tiårene har autonomi gått fra å være en teknologi som tilhører fremtiden til å bli en uunngåelighet. Førerløse biler befinner seg allerede på veiene, og autonome skip ligger godt innenfor rekkevidde. Med en slik økning i autonomi, fører det med seg et behov for ruteplanleggingsmetoder som kan håndtere en stor mengde scenarier. Ettersom de autonome skipene ser seg nødt i å dele havene med bemannede farkoster, er det og nødvendig at ruteplanleggeren følger de marine trafikkreglene som er skissert i "International Regulations for Preventing Collisions at Sea" (COLREGS).

Denne oppgaven undersøker hvordan en stokastisk tilnærming, i form av Path-of-Probability (POP) algoritmen, kan brukes i en ruteplanlegger for å unngå flere statiske og dynamiske hindringer. Videre er gjennomførbarheten til ruten sikret ved hjelp av en simulator, hvor simulatoren består av en fartøysmodell som er utstyrt med et navigerings- og styringssystem. Med disse systemene på plass, brukes en rekke scenarier for å vurdere ruteplanleggeren.

Det ble funnet at sammenføyning av POP-algoritmen med et A*-search var nødvendig for å oppnå endelige resultater, da POP-algoritmen slet i møte med flere statiske hindringer. I kollisjonsunngåelse scenariene viste ruteplanleggeren seg fra en bedre side. Mye av denne forbedringen krediteres til den tilføyede metoden som brukte virtuelle målpunkter i POP-algoritmen. Dette sørget og for at det autonome skipet fulgte COLREGS i tre separate kollisjonsscenarier.

Selv om POP-algoritmen er utsatt for mangler, er dens stokastiske tilnærming fortsatt ansett som en av dens styrker. Dette, ettersom arbeidsmiljøet for autonome skip er stokastisk av natur. Det er derfor viktig å innlemme disse usikkerhetene i en enhver ruteplanlegger.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACE ant colony extended.

ACO ant colony optimization.

ASV autonomous surface vessel.

CPA closest point of approach.

DOF degrees of freedom.

GNC guidance, navigation and control.

LOS line-of-sight.

MPC model predictive control.

NED North-East-Down.

ODE ordinary differential equation.

PDF probability density function.

PID proportional-integral-derivative.

POP Path-of-Probability.

PVO probabilistic velocity obstacle.

SBG set-based guidance.

SDE stochastic differential equation.

VO velocity obstacle.

VTP virtual target point.

Chapter 1

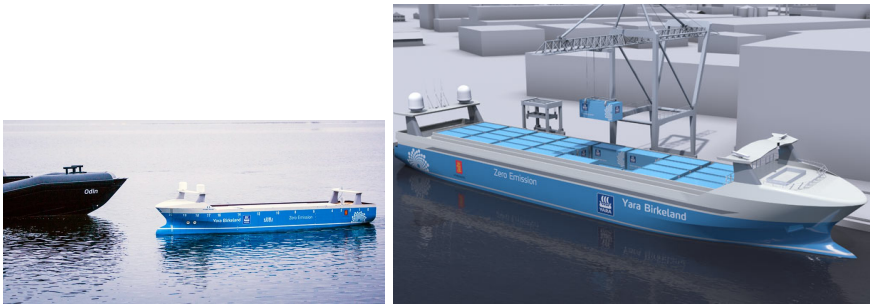
Introduction

1.1 Background and Motivation

In the past decade, autonomy has become an integrated part of our daily lives. Take the robotic vacuums and lawn mowers, which are suddenly considered regular household commodities. An equivalent development is inevitable for both ground and seagoing vehicles, where driver-less cars have already started hitting the roads of California, Paris, Singapore and Beijing. The DARPA Grand Challenge, hosted by the U.S. Department of Defense's research arm, has been a key factor in propelling the level of autonomy forward. Since 2004, the participating autonomous ground vehicles have shown staggering improvements. None of the vehicles managed to successfully navigate through a desert roadway in 2004, whereas four vehicles were able to navigate in an urban environment hardly three years later [1].

At sea, the development of autonomous surface vessels (ASVs) has not followed the rapid pace of the automobile industry. Nevertheless, research is moving in the right direction and large scale ASVs are in the making. Take Yara Birkeland, broadcasted as *"the world's first zero emission, autonomous container feeder"* and set to be fully autonomous by 2020 [4]. Figure 1.1a reveals the current state of the vessel, accentuating that most ASVs are of a smaller scale. Figure 1.1b on the other hand, shows the full-scale version set to be completed in 2020.

The development of ASVs requires a wide array of technologies, spanning from guidance, navigation and control (GNC) systems to sensors. The vessels require sufficient and accurate sensor information to comprehend its surrounding environment. Further, GNC systems must be in place to process the sensor information, generate paths and determine control signals such that the ASV's main objectives are achieved. This thesis will focus on the guidance part of the GNC system, and investigates motion planning for ASVs.



(a) Small-scale version of Yara Birkeland at an autonomous testing land, set to be completed in 2020 [4].

(b) Illustration of the full-scale Yara Birkeland at an autonomous testing site in late 2017 [43].

Figure 1.1: Development of Yara Birkeland, from testing phase to finished product. Credit: Kongsberg.

The ocean presents a dynamic and uncertain environment, enabling a need for robust motion planning methods for ASVs. Further, as ASVs are launched, they will be operating in an environment with other manned vessels. This calls for another dimension of the path planning problem, in that the autonomous vessel must behave in a manner that the manned vessel can interpret and predict. For this reason, adhering to the International Regulations for Preventing Collisions at Sea (COLREGS) is a factor ASVs must consider.

1.2 Research Question and Objectives

This thesis aims to incorporate stochastic theory into the motion planner, subsequently accounting for stochastic effects originating from environmental disturbances, imperfect modelling and uncertainties in the system. It is by means of the Path-of-Probability (POP) algorithm that this task is completed.

At present, the POP algorithm has only been used to solve simple motion planning problems. Further, the algorithm has not been developed for ASVs. In [34], the POP algorithm is used for motion planning of a spherical, rolling robot. In the study, only one *static* obstacle is included in the test scenario. With this in mind, the research question for this thesis is as follows:

> How can the Path-of-Probability (POP) algorithm be enhanced, such that it can generate paths that can avoid *multiple* static obstacles and follow COLREGS in a collision scenario with *dynamic* obstacles?

The specific actions required to answer the aforementioned question are listed below.

1. Design the POP algorithm according to surface vessel models.
2. Incorporate POP in a global path planner[1].
3. Design an algorithm to decide the applicable COLREGS rule in a collision situation.
4. Incorporate POP in a local path planner[2], such that the resulting path follows the traffic rules outlined by COLREGS.
5. Evaluate the performance of the global and local path planners by the means of simulations.

1.3 Main Contributions

The main contributions from this thesis include:

- A detailed literature review concerning the current level of motion planning algorithms for ASVs.
- An alteration of the POP algorithm such that it can generate paths for ASVs.
- A global and a local path planner, where POP has been merged with an A* search to determine feasible paths in an environment with multiple static and dynamic obstacles.
- A feasibility design using the T-neighbourhood from [5], which ensures that the generated path is achievable for the ASV model in question.
- A set of global and local path planning scenarios are created and used to test the motion planning algorithm. Simulation results are provided for each scenario, along with an in depth discussion that pin-points the motion planner's weaknesses caused by the POP algorithm.

1.4 Limitations

The implemented motion planner is developed within this thesis and has not been subjected to testing previously. Therefore, the reader should note that the results obtained here are preliminary results, and the algorithm should be further enhanced in future works. The current limitations of the planning algorithm are outlined below.

[1] Global path planning is explained in section 3.1.1

[2] Local path planning is explained in section 3.1.1, and refers to collision avoidance

In reality, collision avoidance can be achieved via two possibilities. The first is to adjust the velocity of the vessel such that the obstacle is avoided, and the second is to alter the course of the vessel. This thesis will focus on the second approach, where a collision free path is created by altering the course of the vessel. However, the restriction to a constant speed offers a limitation to the local path planner.

The cases used to verify the local planning algorithm are quite simple. First of all, the vessels that are on collision course with the ASV are simulated as non-reactive vessels. This entails that the vessels do not change their course to avoid the possible collision, an aspect that does not mirror practical collision situations. Second of all, the current level of the algorithm does not incorporate the environmental aspects of motion planning. Weather routing [20] is an additional factor that should be included in a sophisticated motion planner, such that the optimality of the resulting path is increased.

1.5 Thesis Outline

The thesis will first take a look at the state of the art methods for motion planning of ASVs in the literature review in chapter 2. Next, important background and theory will be presented in chapter 3, where each section outlines various aspects of the implemented motion planner. As the main contribution of this thesis regards the Path-of-Probability (POP) algorithm and its extension into a dynamic environment, chapter 4 has been dedicated to shed some light on how the algorithm is used for motion planning. In chapter 5, the implementation details for the simulator, global path planner and local path planner are explained in detail. The results and discussion are presented in chapters 6 and 7, whereas the concluding remarks and recommendations for further work are detailed in chapters 8 and 9 respectively.

Chapter 2

Literature Review

This chapter presents a literature review covering various motion planning approaches for ASVs. With the Path-of-Probability (POP) algorithm as the prime focus of this thesis, a literature review is presented to gain more insight as to how POP compares to other motion planning approaches. It aims to take a holistic view on some of the state of the art methods used for motion planning of ASVs. The review is an abbreviated version of the literature review presented in [17], which is the project thesis used as a preparation for this master's thesis.

Solving the problem of safe motion planning for ASVs is a multi-criteria optimization problem. In other words, it is a complex task that can be solved by numerous methods. The available research on the topic mirrors this, presenting a variety of approaches rooted in different methodological areas. With this in mind, this literature review will take a methodological approach to categorize the methods. The focus will be on the following three approaches: stochastic, deterministic and computer intelligence, as shown by figure 2.1. Before the review of each method, the common assumptions used in research for motion planning are outlined in section 2.1.

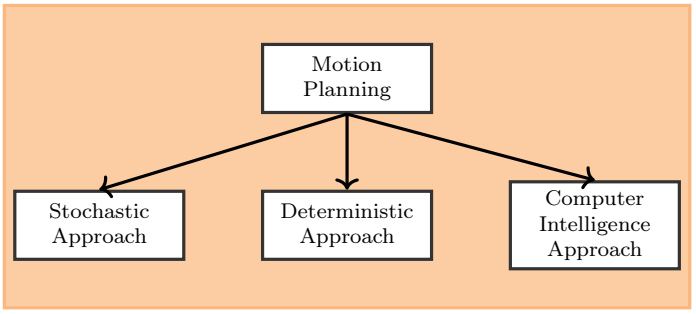


Figure 2.1: Division of the various approaches to motion planning

The reader should note that the methods for global and local motion planning[1] are reviewed simultaneously, such that there is no division between algorithms that may be suited for collision avoidance or global planning. This set-up is justified as global and local planning both have the same objective,; to determine a safe and feasible path for the ASV. For the remainder of this chapter, the vessel that is subjected to control will be referred to as the *own vessel* (OV), whereas the vessels that are treated as obstacles are called *target vessel* (TV). According to COLREGS terminology, in a collision situation there is a *stand-on vessel* and a *give-way vessel*. The stand-on vessel is set to follow its current path without the use of evasive maneuvers, whereas the give-way vessel is set to *"keep out of the way"* of the other vessel and *"keep well clear"* [25].

2.1 Common Assumptions

In motion planning, it is often assumed that the navigation system in figure 3.1 provides the vessel with satisfactory information regarding its own behaviour, i.e its position, orientation and velocities. Further, sufficient data information regarding the surrounding environment and other vessels is also assumed, such that the guidance system has an accurate view of the environment when the path is planned and re-planned.

A second common assumption that unites the various collision avoidance methods, is that the own vessel carries out the evasive maneuver to avoid collision; implying that the target vessel(s) does not change its course or speed when a collision risk is present. Keeping in line with the COLREGS terminology, this means that the own vessel will typically act as the give-way vessel in the collision scenarios. This assumption is valid in most cases, as the aim of the research is often to determine how the guidance system of the ASV is able to cope with a collision situation. However, in a real scenario, the target vessels are likely to carry out evasive maneuvers. This is especially important for situations with multiple target vessels present.

2.2 Stochastic Approaches

A human navigator will use his/her intuition and continuously update the current trajectory such that the probability of reaching the desired position is maximized. This also applies in a collision situation, where a human navigator will use an intuition based on probability to determine an evasive trajectory that is least likely to collide with a target vessel. Stochastic approaches to motion planning for ASVs try to mimic this behaviour.

The main contribution from stochastic methods, compared to deterministic methods, is that they address the uncertainties that originate from the surrounding

[1] The definition of global and local motion planning is outlined in section 3.1.1

environment, including uncertain target vessel motions and uncertainties in the own vessel motions caused by environmental factors; and sensor measurements. In recent research, probabilistic approaches have been used for handling and detecting obstacles but also as a basis for the choice of a trajectory. We will first take a look at how probability has been used to track obstacles and then look into how it has shaped planning algorithms.

2.2.1 Handling Obstacles with Probability

The probabilistic obstacle handling introduced in [6] allows the guidance system to closely follow the behaviour of an experienced navigator, in that the own vessel will stay on its initial path until the probability of a collision exceeds a given threshold. Only then will the collision avoidance module be triggered, and the own vessel will perform an evasive trajectory. Thus, resulting in a reduced number of evasive maneuvers completed. The guidance system in [6] goes about predicting a target vessel's position using a modified version of the *constant velocity* (CV) and *constant turn rate and velocity* (CTRV) models. Instead of assuming that the estimate of the target vessel's position is certain, as is the case for the deterministic version of the CV model in equation (2.1), the modified version in equation (2.2) has an added random variable ϵ_t to account for the unpredictability of a target vessel's position.

$$x_{t+T} = x_t + \dot{x}_t T \tag{2.1}$$

$$x_{t+T} = x_t + \dot{x}_t T + \epsilon_t \tag{2.2}$$

Using these stochastic models, [6] creates an occupancy grid over the state-space that maps the obstacles; each cell in the grid has a probability of containing an obstacle. In a collision situation, the evasive trajectory is found by completing an A*-search over a two dimensional grid (x,y) of cells in the state-space. The occupancy grid guides the search by applying a penalty to the cells with a high probability of containing an obstacle. Resulting in a set of waypoints that are least likely to collide with the target vessel.

2.2.2 Motion Planning with Probability

Unlike [6], where stochastic models are used in the preliminaries to the actual motion planning, [34, 30] use stochastic approaches to choose the path that is most likely to reach the target. [34] uses a method called the Path-of-Probability (POP) algorithm, whereas [30] uses a version of the velocity obstacle (VO) method called probabilistic velocity obstacle (PVO). First, PVO is examined, followed by a more detailed look at POP's role in research.

Probabilistic Velocity Obstacle

PVO stems from the popular velocity obstacle (VO) method, which is a motion planning method that has been used on unmanned aerial vehicles (UAVs) in [12] and on ASVs in [51]. The gist of VO lies in defining a set of relative velocities that will ultimately lead to a collision. This velocity space, which is often referred to as a collision cone, represents a geometric region that the mobile robot must not enter to avoid a collision. VO is explained in further detail in section 2.3.1.

[30] points out that a drawback to the VO algorithm is that it typically assumes that the moving obstacles have no perceptions or motion goals. So, if mobile robots are to successfully navigate in an environment with for instance humans, that will perceive the mobile robot and may adjust their velocity therein, it must account for uncertainties in the obstacle velocities. This is what [30] achieves by introducing the probabilistic velocity obstacle (PVO) method, where the uncertainties related to object tracking techniques are addressed. PVO differs from VO in that the collision cone is defined according to the probability of colliding with an obstacle, which includes the uncertainty in the velocity measurement of the obstacle. The addition of a probabilistic approach to VO means that the motion planner now has two objectives: reaching a goal *and* minimizing the probability of a collision.

These two objectives are reached by maximizing a relative utility factor, called RU_i in [30] and shown in equation (2.3). Here U_i is a function representing the utility of velocities v_i for a set goal, D_i represents the dynamic feasibility of v_i and PVO_i represents the PVO function for the same velocities. From the definition of RU_i, it is clear that by navigating with PVO by maximizing RU_i, we are minimizing the value of PVO_i. This means that the PVO algorithm will choose velocities that are *least* likely to collide with obstacles, making the PVO method a stochastic approach to motion planning.

$$RU_i = U_i \cdot D_i \cdot (1 - PVO_i) \tag{2.3}$$

Path-of-Probability

Path-of-Probability (POP) was first introduced to solve the inverse kinematics problem of discretely actuated manipulators in [16]. Since then, it has been developed to work for a variety of circumstances; ranging from motion planning of flexible needles in medical applications in [44, 3] to path planning for rolling robots in [34, 35].

The method is based on models of stochastic differential equations and their solutions. For each intermediate step, the next point in the path, shown as g_i in figure 2.2, is chosen such that the probability of reaching the target position g_{goal} is maximized. In figure 2.2, the optimal choice for g_i with respect to probability is the red path, where the probability of reaching the goal is larger than if choosing

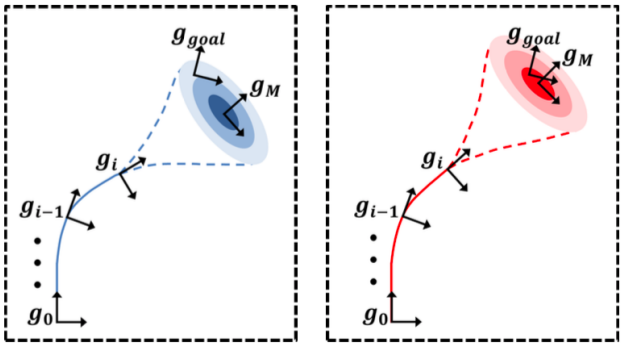


Figure 2.2: Illustration of POP algorithm from [34]. The candidate point g_i in the blue path is less likely to reach the target point g_{goal}, as illustrated by the surrounding probability density function with center in g_m.

g_i in the blue path. The details of the algorithm are explained in greater detail in section 4.1.

In [34], Lee and Park first confirm that the rolling robot exhibits uncertain behaviour and then create a stochastic model to simulate the rolling robot's motion. To ensure feasibility of the resulting path from the POP algorithm, [34] discretizes the input velocities to obtain a set of candidate points. This discretization limits the algorithm, in that each intermediate path must be selected from a finite set of candidate points originating from a pre-defined set of discrete inputs. This discretization means that there is a chance that the final path will fail to reach the target position. However, [35] addresses this limitation by using fuzzy logic to transform the discrete inputs into a set of continuous candidate points. [35] proves that this approach greatly improves the POP algorithm by reducing the roughness of the final path and obtaining a global path closer to the target position. This approach shows that the POP algorithm is adaptable to fit with other methods in order to overcome its drawbacks.

As for collision avoidance with POP, research only shows to simple collision situations; such as avoiding a single static obstacle. Even though [34] show that the algorithm is successful, the scenario is too simple to conclude that the POP algorithm can efficiently handle all types of collision scenarios. Therefore, more research is needed on this topic.

2.3 Deterministic Approaches

A deterministic model is used to indicate how the vessel *will* behave whereas stochastic models predict how the vessel is *likely* to behave. Deterministic models will therefore typically not account for the uncertainty in each output from the

model. In this section, methods for motion planning that do not directly consider the uncertainty in the vessel's states, both the own vessel and target vessel, will be discussed.

2.3.1 Collision Avoidance with Velocity Obstacle

The velocity obstacle (VO) method is used in collision avoidance maneuvers, i.e local path planning, and constructs a set of relative velocities between the own vessel and the target vessels that will ultimately lead to a collision. This velocity space, referred to as a collision cone by [19], sets up a geometric region in the shape of a cone that signifies a risk of collision. This cone is can be seen in figure 2.3. If the vessel is to avoid a collision with the target vessel, it must select a velocity that lies outside the collision cone and then steer its course to follow the chosen velocity. From figure 2.3, the own vessel (A) must choose a velocity that does not lie within the collision cone, i.e the area shaded in purple ($VO_{A|B}$). This guarantees that a collision-free path is chosen, though under the assumption that the target vessel maintains its current shape and speed [19].

VO subtly ensures that the avoidance maneuver is dynamically feasible by intersecting the set of achievable velocities with the set of avoidance velocities [19], which is the set of velocities outside the collision cone. In the generation of the achievable velocities, the actuator dynamics are considered alongside the feasible accelerations of the vessel. Incorporating the imperfections in the actuator in the collision avoidance is a strategy that many motion planning algorithms neglect, whereas VO considers it effortlessly. This simple and straightforward approach is one of the strengths of the VO method. However, as [39] points out, VO assumes linear and constant velocities to predict the target vessels behaviour. This assumption may be invalid, especially if the target vessel also responds to the own vessel's presence and performs an evasive maneuver to avoid collisions. Noteably, this issue has been addressed by [30], with the probabilistic velocity obstacle (PVO) method. However, the simplicity of the VO algorithm renders the results somewhat unrealistic. Further, relying on models in this manner also signifies one of the drawbacks of the method. As inconsistencies or inaccuracies in the model will lead to unfavourable results, which could end up causing a collision.

In order to create a safe and feasible trajectory, a sequence of avoidance maneuvers must be selected based on the VO algorithm. This can be completed by searching over a tree of feasible maneuvers at discrete time intervals [19]. In order to reduce the computational complexity of the method, which is especially important for on-line applications such as reactive avoidance collision, heuristics can be used to reduce the search space. These heuristics can be designed to include multiple criteria, thus the method can be used to allow for optimization based on a wide variety of criteria. Another option to reduce the computational complexity is by discretizing the velocity field and then using a cost-function for each of the possible finite velocities. [51] designed a cost-function such that COLREGS compliance was one of the criteria for choosing the velocity. VO worked well with COLREGS

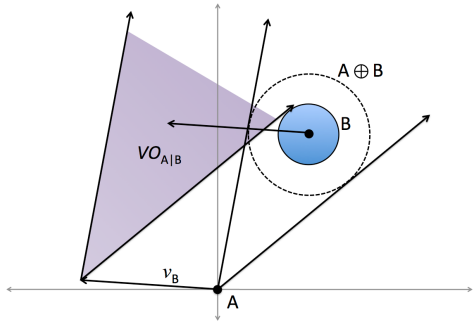


Figure 2.3: Visualization of the collision cone in the Velocity Obstacle collision avoidance algorithm, where A is the own vessel and B is the target vessel. Image is obtained from [29].

according to [51], although the simple use of a Boolean operator for COLREGS compliance in the cost function made the method struggle once the vessel was in between COLREGS rules. Still, the amount of research on VO in motion planning reflects that the algorithm shows promise.

2.3.2 Switching between Set-Based Tasks with Set-Based Guidance

The set-based guidance (SBG) approach is a more holistic approach to designing a motion planning system for ASVs, as it allows for any combination of guidance laws in path following *and* collision avoidance. Therefore, SBG does not have a specific path planning or collision avoidance approach, rather it provides a strategy for how the motion planner can be split between global and local path planning. The heart of the method lies in the switching mechanism, where the planner switches between a path following mode and a collision avoidance mode once a certain criteria is satisfied. SBG suits the collision avoidance problem for ASVs as collision avoidance can be considered as a set-based task [40], where the collision avoidance task is prioritized over the global path following task.

Set-based tasks are tasks that are valid within an *interval* of values rather than just at an exact value, meaning that the switching between the path following mode and the collision avoidance mode occurs once a defined criteria variable is reached. [41] experimentally proved the use of SBG and set-based tasks by solving the inverse kinematics problem of a 6 DOF robotic arm. The switching mechanism between the pre-defined tasks in [41], described by $\boldsymbol{\sigma}(\boldsymbol{q})$ where $\mathbf{q}$ is the joint angles of the robotic arm, differ from the switching mechanism for a motion planner of an ASV.

The extension from the inverse kinematics problem to motion planning for ASVs has been investigated by, among others, [40] and [42]. Their switching mechanism σ, shown in equation (2.4), calculates the distance between the own vessel and the

target vessel. Here it is assumed that the position and behaviour of the obstacle is perfectly known. It is in this step that the stochastic models would typically implement an uncertainty, such that the switching criteria σ becomes a stochastic variable. Therefore, to represent the real world uncertainties in sensors and mathematical models, the method would have to include some notion of uncertainty. However, [40] and [42] aim to extend SBG from fully actuated vehicles to underactuated vehicles operating in a dynamic environment, not to extend the method to incorporate uncertainty.

$$\sigma = \sqrt{(x - x_o)^2 + (y - y_o)^2} \tag{2.4}$$

In the event that an obstacle is detected, the implemented collision avoidance mode will be triggered once the ASV reaches a safety radius around the obstacle. However, as the ASV is not supposed to enter the safety radius, the collision avoidance mode must be triggered *before* the ASV's position intersects with the safety radius. Thus, [42] uses two safety radii; one that contains the forbidden area and on that triggers obstacle avoidance. The resulting collision avoidance behaviour from the ASV is convincing, [40] proves how the ASV successfully avoids three vessels moving in different directions. Also the smooth convergence to the original straight path is impressive. According to [40], the method is robust, as the own vessel was able to avoid collision following COLREGS even though the target vessels did not follow the rules. [42] brings forth how SBG is not able to handle multiple obstacles in close proximity to each other, as it runs a risk of getting stuck between them. This drawback originates from the collision avoidance method used, where both [42] and [40] track a safe radius around each obstacle in order to avoid them. It is therefore possible to extend the SBG system such that it can handle overlapping obstacles.

2.3.3 Motion Planning with Model Predictive Control

model predictive control (MPC) is an example of a method that relies on deterministic models yet addresses uncertainties in the prediction of the obstacles' trajectories. It has been used for collision avoidance for ships by [27] and [22], and has been developed for controlling autonomous cars by [47]. The method is, in its essence, based on using a range of control methods that explicitly use a process model to determine which control signal minimizes an objective function [9]. This can be translated into collision avoidance, where the control inputs that produce the optimal trajectory with the smallest hazard are chosen as the final control action.

MPC is quite computationally complex, involving numerical optimization techniques that may fall into local minima, such that the true minimum value for the optimization problem is not found. In order to veer from these issues, [27] presents a simplified version of the method. *Simulation-based* MPC uses a ship model to simulate the resulting trajectories of the ship based on varying control inputs. [27]

has reduced the set of possible control inputs to a discrete set, where the only parameters that can be varied are the nominal speed and the course offset. The resulting trajectory of each control input is then evaluated in a simulated environment over a finite prediction horizon. The hazard of each trajectory is evaluated with a cost function, where [27] accounts for COLREGS compliance, risk of collision and penalizes additional control effort to alter the trajectory. The cost function allows for multiple constraints to be regarded in the optimization problem; highlighting the flexibility of the method.

[27] shows that with a few alterations, MPC is able to solve the path planning problem with a low computational time. Further, the incorporation of COLREGS compliance in the cost function is effective, as results show how an additional cost due to violation of a COLREGS rule reduce the optimality of a trajectory and leads to it being discarded. The method is also shown to handle multiple dynamic obstacles. The credibility of the results is high, with as much as 12 different scenarios evaluated with multiple obstacles. One of these scenarios incorporates environmental disturbances in the form of wind and current forces acting on the vehicle, [27] presents a disturbance-sensitive algorithm that is able to predict how drift will affect the trajectory. By using a standard 3 DOF model of the own vessel, wind and ocean forces are incorporate in the kinematic equation and in the vehicle's equation of motion in the BODY-fixed reference frame. As MPC finds the optimal speed and course offset based on simulations of a deterministic model, the dynamic feasibility of the chosen trajectory is then secured. Although, using an incorrect model will obviously lead to trajectories that may not be feasible. Hence, the method suffers from some lack of robustness as an accurate model is necessary. The modelling of the obstacle's trajectories however, stand out. As the uncertainties of using a straight line prediction is considered by applying the method to a *set* of velocities and bearings, rather than one constant value for the velocity and bearing for each obstacle. [27] then incorporates a degree of uncertainty without the use direct use of stochastic models. All in all, the results with simulation-based MPC are impressive and show promise.

2.4 Computer Intelligence Approaches

Computer intelligence, commonly referred to as *soft computing*, is a broad topic with no clear definition. According to [57], the main difference between hard and soft computing is their tolerance of precision and certainty. The traditional hard computing methods rigorously value precision and certainty in their outcome. Soft computing on the other hand, mimics the human brain in its reasoning and computation, as the difficulty of complex problems is reduced by softening the degree of precision and certainty of the outcome. The human brain is able to rationalize and make decisions in uncertain and chaotic environments. Take deciphering handwriting, we encounter numerous ways of writing each letter, yet we are able to read and fully comprehend most handwritten texts. If a computer were to under-

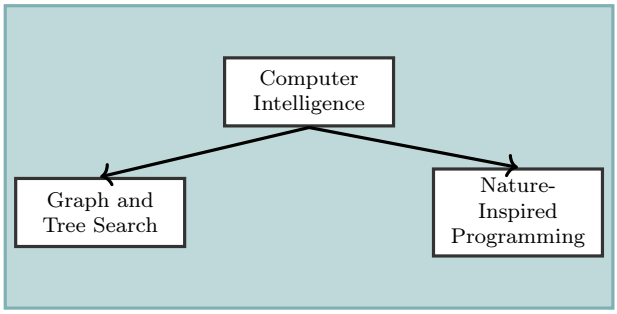


Figure 2.4: The two approaches within computer intelligence that will be covered in this review.

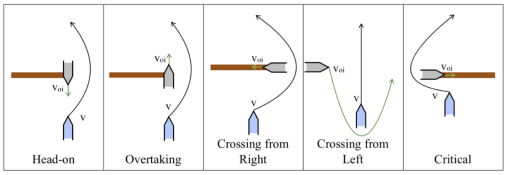
stand handwritten notes, it would have to reduce its constraints on the precision of classifying, say the letter "a". This is where soft computing enters, and makes computer intelligence imitate human intelligence.

Figure 2.4 shows the different approaches within computer intelligence that will be covered in this literature review. In the sections below, their relation to path following and collision avoidance will be presented. Note that the various approaches to computer intelligence span much wider than that given by the overview in figure 2.4.

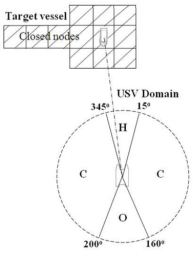
2.4.1 Graph and Search Tree Algorithms

In this section the popular A* search algorithm will be covered, which falls under the tree search category. It has typically been used to plan paths for mobile robots based on a known and static environment, although it has been extended to ASVs operating in a dynamic environment. One such example, [6], has been reviewed in section 2.2.1. According to [10], this heuristic approach is well suited for motion planning for marine crafts as the problem will be two-dimensional in nature. A* motion planning has been introduced by [11], where a suitable collision avoidance maneuver is obtained by searching over a map that includes dynamic obstacles. This method has been extended by [10] to include COLREGS compatibility, resulting in a heuristic Rule-based Repairing A* (R-RA*) algorithm. Further, the notion of target tracking using the A* algorithm is introduced in [2]. These two papers, [2, 10], will be investigated in further detail below.

The main objective in [2] is to track a marine craft autonomously all the while evading dynamic obstacles. In order to set up a search space suitable for the A* algorithm, the motion of the target vessel must first be predicted. This is completed using Monte Carlo sampling, where fuzzy weights are used to give the trajectories that follow human-like steering a larger probability. The Monte Carlo sampling is based on perfect knowledge of the target vessel's current states. However, the uncertainty regarding the target vessel's future behaviour is addressed in the out-



(a) The brown indicates the artificial terrain costs added to the search space, ensuring that the resulting evasive maneuver in [2] satisfies COLREGS.



(b) USV domain split into various segments to classify which COLREGS rule applies to the approaching target vessel, this approach is used in [10].

Figure 2.5: Two different approaches for incorporating COLREGS compliant behaviour using A* search.

puts of the Monte Carlo sampling, as it outputs the most probable path the target vessel is likely to take. Thus, [2]'s approach with the A* search is aware of the uncertainties related to tracking target vessels. Further, the next move in the A* iterative search is chosen under a constraint such that it is in line with a 3 DOF dynamical model, as specified in [20]. This then secures dynamic feasibility of the resulting path to follow the target vessel. As for the computational complexity of the method, it will depend upon the method used for estimating the target's trajectory and the size of the search space for the A* algorithm. [2] mentions how the computational time increases when using Monte Carlo sampling rather than the traditional target tracking methods, such as pure-pursuit and constant-bearing. However, with Monte Carlo sampling the resulting estimation of the target's trajectory is shorter and thus minimizes the time the own vessel uses to catch up with the target. The results in [2] show that the A* motion planning successfully follows the trajectory of the target vessel in the presence of dynamic obstacles in a simulated environment. Also, experimental results are used to show the validity of the procedure, and how it is able to operate in a real environment.

The COLREGS compliance of the resulting maneuver is secured in both [2] and [10], though the two approaches differ. As figure 2.5a shows, [2] merely adds an artificial terrain cost to the paths that will disobey COLREGS. This means that the A* search will deem these path as less suitable, and therefore not choose them. The approach in [10], is to split the domain of the USV into sectors that represent the various COLREGS situations. Suppose a target vessel approaches the USV at an angle between 15-45°, as shown in the example in figure 2.5b (assuming there is a mistake in the graphics from [10] where 354° is supposed to be 45°). Then the paths that disobey COLREGS, such as the paths going to the starboard side of the target vessel in a head-on situation, will not be searched over. This is done by adding the paths that go through the closed nodes in figure 2.5b to the closed list

in the A* search.

Unlike [2], the objective of the path planning algorithm in [10] is to extended the regular A* algorithm such that it can account for COLREGS in a collision situation. This is completed by introducing the R-RA* algorithm, which will give a COLREGS compatible path. The way in which this is completed is described above and shown in figure 2.5b. The collision avoidance module will first create a global path from the initial position to the goal position, completed offline. Next, the R-RA* algorithm searches through a path segment of this global path, and then checks for possible target vessels. If a target vessel is present, then its velocity is determined, and the decision maker assesses the COLREGS situation and adds the paths that are out of bounds to the closed list. This step involving R-RA* is completed for each time-step that the map is updated. The resulting evasive maneuver will be given by a set of waypoints. As for the uncertainty of the target vessel's position, [10] does not consider it. However, it was shown in [6] how probabilistic obstacle handling could be merged with an A* search to find an optimal evasive trajectory. Thus, there are possibilities in research to account for obstacle and environmental uncertainties.

2.4.2 Nature-Inspired Programming

Nature-inspired programming is a kind of computational intelligence that uses biological processes to model and solve global optimization problems. The various methods are further categorized in figure 2.6, where a distinction is made between *swarm intelligence* and *evolutionary computation*. The two methods fall under the population-based *metaheuristic* category, in that they use problem-independent algorithms to approximate a solution to the global optimization problem. This approach contrasts to *heuristics*, in that the metaheuristics algorithms do not have to deeply adapt to each problem at hand [7]. As a result, metaheuristics is often used to solve hard optimization problems, such as motion planning in a dynamic environment.

Both of the aforementioned approaches mimic nature's way of continuously optimizing by evolution and selection, though swarm intelligence is more related to nature's collective intelligence than to its evolutionary process. The evolutionary computation algorithms are, as the name suggests, more closely related to optimization through evolution. Only the fittest survive in nature's evolution through selection, which in an optimization problem translates to that only the most optimal solutions proceed to the next round of the solution process.

Ant Colony Optimization

ant colony optimization (ACO) is a kind of swarm intelligence approach, and has been used for motion planning for marine crafts. ACO has been studied extensively

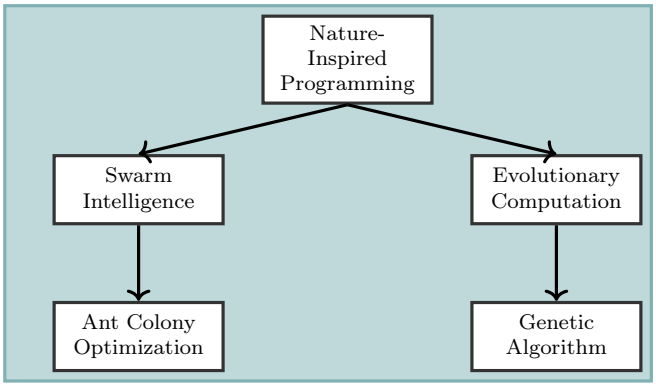


Figure 2.6: A further division of the approaches within nature-inspired programming that are covered in this paper

in the path planning problem for ships, [53, 32, 18] all use versions of ACO to find the optimal collision avoidance strategy for an ASV.

Lazarowska has presented numerous studies of how to integrate ACO into collision avoidance and trajectory planning for a ship. In the papers [32, 33], the own vessel is able to successfully navigate in an environment with static and dynamic obstacles, where some of the simulations were run with as much as 12 target vessels. However, Lazarowska does not account for uncertainties, in that the behaviours of the target vessels are perfectly known and there are no environmental disturbances in the results. This limits the credibility of the results, and should be accounted for in further research. Although the computational complexity and real-time implementation of the trajectory planning with ACO shows a brighter side of the research. With a limit on the computational time of 60 seconds, the procedure is close to determining safe paths in real-time. However, as simple models are used in [32], the computational complexity will be reduced. This allows the procedure to find a sub-optimal solution in less time than if more accurate models were used. Therefore, it is predicted that an implementation accounting for these missing factors would suffer from a longer computational time. Further, the dynamic feasibility of the study is limited, as the feasibility of a constructed path is determined by simply checking the time of the intended maneuver. It is not guaranteed that a vessel will be able to follow a specific maneuver to avoid dynamic obstacles, even though it may get from one waypoint to the next in a specified amount of time. Take for instance a sharp turn, a large vessel is limited in its turning radius.

An extension of ACO for motion planning for marine crafts is presented in [18], where a new algorithm called ant colony extended (ACE) is used to find a sub-optimal path. The aim of this research is to investigate how ACE improves the flaws of ACO in motion planning, which include falling into local minima and a lowered dynamical feasibility of the search space. Therefore, [18] merely simulates the algorithm with static obstacles.

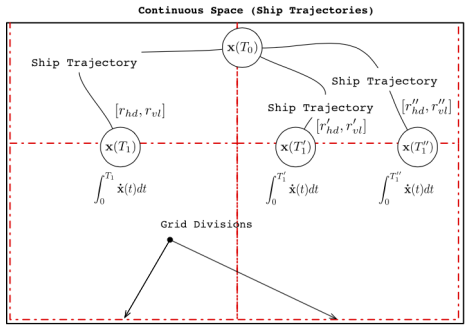


Figure 2.7: The discrete cell-like environment with the continuous state-space of the vessel. Image collected from [18].

Unlike ACO, ACE does not search for solutions through a node-like search space. Rather, it creates a discretized map consisting of *cells*. Within each cell, the ants will choose a heading and speed that will remain constant within that cell. Thus, resulting in different trajectories for each choice of heading and speed, as illustrated in figure 2.7. A discrete set of possible set-points for the heading and speed are set, and the ants choose a heading and speed either using heuristics, previous knowledge or an existing pheromone trail. The heuristic choice is used when there is no prior knowledge, where the heuristic for the heading accounts for the vessel dynamics ensuring that the constructed trajectory is dynamically feasible. As a result, the ants are not searching for the optimal route along fixed nodes but rather searching for the optimal sequence of set-points to go from the initial state to the target state. This increases the feasibility of the resulting trajectory compared to the approach in [32]. As for the local minima issue, ACE disregards a solution only if it exceeds a certain time-limit. According to [18], this technique will ensure that the algorithm will not get stuck in local minima. The simulation results presented are convincing, as the algorithm successfully maneuvers in an environment cluttered with static obstacles. However, the simulation environment is a simplified version of reality as no environmental disturbances and dynamic obstacles are incorporated.

Genetic Algorithm

The use of genetic algorithms to motion planning was first introduced in 1994 by [37], applying the techniques to mobile robots. For surface vessels, [48] first introduced the use of evolutionary algorithms for trajectory planning. Since then the use of evolutionary computing has increased and gained popularity, as the level of research shows. In this section, three examples of how evolutionary computing has been used in solving the path planning problem for ASVs will be investigated.

The first use of evolutionary computing comes in the form of a genetic algorithm in [54], where an assistant navigator is created. By assistant, it means that the algorithm will find a solution and then propose the evasive maneuver to a human navigator. This then reduces the need for understanding the solutions originating from the genetic algorithm, as a human operator will interpret and decide if the resulting path is safe. [54] chooses to optimize the new path with regard to an economic viewpoint, where the distance of the new path and the deviation from

the original path is minimized within the optimization problem. As a result, the fitness function in [54] depends on these two variables, and only a course change is used to plan the collision avoidance route. When that is said, [54] assumes perfect knowledge in tracking the target vessel's position, velocity and attitude. Hence, uncertainties in observational measurements are not accounted for. As for COLREGS compliance, [54] runs simulations to mirror COLREGS situations. However, the way in which the genetic algorithm adheres to COLREGS is not detailed within the paper. Though the resulting simulations, that notably only include one dynamic obstacle, manage to adhere to the COLREGS rules. Further, the dynamic feasibility of the resulting trajectory is secured. This, as the course changes to perform the evasive maneuver are minimized in the algorithm and will most likely be within a possible turning range for the vessel.

The second use of evolutionary computing is in [14], where an evolution-based path planning system is introduced to plan missions for multiple ASVs. [54] uses a binary gene coding to represent each path within the population, whereas [14] has a population consisting of waypoints used to represent a possible path. This means that in the mixing step of the algorithm, the most fit paths are either mutated or reconfigured to create new paths that are evaluated by a fitness function. The dynamical feasibility of the resulting path from the waypoints is secured, as [14] includes constraints in the fitness function to represent the vehicle's turning rate constraints. An increase in path feasibility could have been introduced by using the vessel dynamics as constraints as well, although [14] points out that the vessel will not make aggressive maneuvers thus only incorporating the turning rate is justified. Further, a reduced fitness is given to the paths that disobey COLREGS, resulting in simulations that show COLREGS compliance. Rather than projecting the behaviour of the target vessels ahead in time, the evolution-based path planner regularly updates the planned path based on sensor information. If this interval is short enough, and the sensor information is accurate, then the uncertainty regarding the final path and the positions of the target vessels will be reduced. Although, re-planning the path in small time intervals will increase the computational time of the algorithm. However, the resulting simulations show that the path planner based on an evolutionary approach is able to successfully navigate in a multi-dynamic vehicle environment.

Finally, the use of NSGA-II to solve the motion planning problem is presented. This is an extension of the genetic algorithm, called non-dominated sorting genetic algorithm (NSGA). [56] presents a version of NSGA, called NSGA-II, which is commonly used for multi-objective optimization. Thus, making it suitable for the multi-objective optimization problem that is ASV motion planning. In [56], the optimization is carried out to find the optimal rudder angle that results in the optimal trajectory, taking a different approach than [14, 54]. Like [54], [56] incorporates economical and safety factors to evaluate the fitness of the solutions. Specifically, [56] has three objectives, (i) security, (ii) economic factor and (iii) smoothness factor. The Nomoto model is used to simulate the yaw motion caused by a change in the rudder angle, the resulting trajectory is then evaluated for safety

with regard to the target vessels. The use of a Nomoto model, will ensure that the evaluation of the paths from various rudder angles reflect the vehicle's dynamics. However, the accuracy will depend upon the degree and parameters within the Nomoto model. Also, incorporating a smoothness factor ensures that rudder commands that give non-smooth trajectories will be given a reduced fitness value, and thus means that their chance of being used to create offspring is limited. With this in mind, the dynamic feasibility of the resulting path will most likely secured. In [56], COLREGS compliance is also considered, and is included in the fitness function under the safety criteria. Though, the research presented by [56] only considers the mechanism of using NSGA-II for collision avoidance. Thus, further work should be completed where sensor uncertainties, environmental disturbances and multiple targets are considered. Yet, the preliminary results are promising.

The examples mentioned above show the versatility of the genetic algorithms. However, the method has its disadvantages. The mutation step within the algorithm introduces a randomness to the solution, as a random solution from the parents is mutated to create a new solution in the offspring set. This makes the algorithm a non-deterministic method and thus results in different results for the optimal solution for each run. For motion planning of ASVs, this can be a drawback. If human operators are to rely on an autonomous navigation system, then consistency of the results and an understanding of how the results come about is important. However, the issue of falling into local minima is somewhat reduced by using evolutionary computation. This reduction stems from the use of a population of solutions rather than the classical approach of maintaining a single best solution found so far. This will then increase the chances of finding global optimum solution [21]. Finally, genetic algorithms have a trade-off between optimality and computational time. A larger population in each generation means there is a larger span of solutions, and thus increases the chance of obtaining an optimal solution. Yet, larger populations increase the computational time.

2.5 Summary of Reviewed Work

This literature review presented an outtake of motion planning methods for ASVs, including deterministic, stochastic and computer intelligence approaches. Each approach has its advantages and disadvantages.

Take deterministic models, these are often quite accurate but rely on complex models that increase computational time. If ASVs are to operate at sea, they must be able to quickly respond to a collision situation, making it preferable that the methods can be completed in *real-time*. This is where computation intelligence enters. Rather than using complex models to mimic reality, algorithms that mimic nature's selective behaviour are used. Further, the working environment of the ASVs constitutes stochastic variables, in that oceans, weather and the behaviour of other vessels is not certainly known. Motion planning methods that rely on stochastic models try to handle this issue, where the optimal collision avoidance

behaviour of the ASV is selected based on the most-probable optimal behaviour. The stochastic approaches are more fit for practical use than their deterministic counterparts, in that it is important to incorporate the environmental and sensor uncertainties.

In research, all the methods show promising results. Although, incorporating the uncertainties in sensor measurements and the surrounding environment is an important criteria for the realization of the ASVs. Further, the computational time for re-planning a path must be close to real-time if the ASVs are to be considered safe. With these two criteria in mind, a combination of the stochastic models and the computer intelligence algorithms stand out as the better choice. However, if accuracy is valued highly, the mathematical models will stand out. Also, as computational power increases, the possibility of employing complex models in an optimization problems seems more attainable. Currently, more research is needed. Yet, based on the insights gained in this literature review, the research looks promising.

Chapter 3

Background and Theory

This chapter presents relevant theory for this thesis' main contribution, namely motion planning. Further, some background information regarding guidance, navigation and control (GNC) systems and COLREGS is also introduced, as these are crucial for the simulation environment and an overall understanding of autonomous surface vessel (ASV)s.

3.1 Guidance, Navigation and Control systems

GNC is used to describe systems that automatically control moving vehicles [20]. If autonomous vehicles are to safely navigate in their respective environments, one such system is needed. This section will cover GNC systems for a marine craft, illustrated in figure 3.1, and will take a deeper look into a marine craft's guidance system; as that is the main focal point of this thesis. For the remainder of this chapter "craft" will be used to refer to a marine craft. But first, some terminology regarding motion planning is covered.

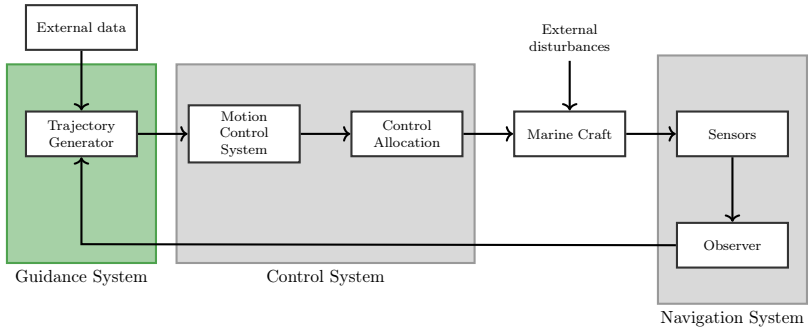


Figure 3.1: GNC system.

3.1.1 Motion Planning Terminology

This section considers some of the terminology used in motion planning, where the various kinds of motion planning and the terminology used in collision avoidance is covered.

Global and Local Motion Planning

The motion planning problem for ASVs can be divided into two sub-problems, *global* path planning and *local* path planning. Local and global algorithms differ in the area that the guidance system considers.

Global path planning refers to the motion planning that is completed in a fixed and known environment, where the path is generated from the vessel's initial position to its target position. With this global perspective, static obstacles are accounted for. Dynamic obstacles on the other hand, are not considered as the locations of these obstacles are subjected to uncertainties. The distance covered in global path planning is typically quite large, which entails that the estimated locations of the dynamic obstacles is subjected to correspondingly large uncertainties. Thus, including dynamic obstacles in the global path planner becomes futile. This is where local path planning enters.

Local path planning considers a bounded area around the ASV, typically in the range of the sensor measurements. This means that the local algorithm will update its view of the surrounding environment and plan the route thereof. It is often used for collision avoidance, and will therefore be referred to as such throughout this thesis. The task of the local path planner is to account for unforeseen events in the global path, in that the global path may collide with an approaching dynamic obstacle.

Own Vessel and Target Vessel

In a collision scenario between two vessels, a distinction between the *own vessel* (OV) and the *target vessel* (TV) is made.

The vessel that is subjected to control is called the own vessel, and will be referred to as the "ASV" throughout this thesis as the aim is to control an ASV in a static and dynamic environment. Alternatively, the vessels that are treated as obstacles are called target vessels. According to COLREGS terminology, in a collision situation there is a stand-on vessel and a give-way vessel. The stand-on vessel is set to follow its current path without the use of evasive maneuvers, whereas the give-way vessel is set to *"keep out of the way"* of the other vessel and *"keep well clear"* [25].

3.1.2 Guidance Systems

The implementation of the guidance system will depend upon the craft's objective. Were it to track another moving craft, follow a pre-determined path or converge to a fixed set-point; then the guidance laws would differ. According to [20], these objectives can be classified into the following categories: *trajectory tracking*, *path following* and *setpoint regulation*. The main difference between the three lies in the constraints, where both path following and trajectory tracking have spatial constraints but trajectory tracking has added temporal constraints. Setpoint regulation has the simplest constraints, in that it concerns maintaining constant target positions and orientations.

No matter the objective, the guidance system's goal is to determine a safe trajectory for the vessel. This problem can be extended to a multi-objective optimization problem, where the different methods used in research are discussed in chapter 2. There are numerous criteria that may be considered in the optimization problem; examples include time, fuel efficiency, COLREGS compliance, safety, and so on. Nevertheless, once the trajectory is determined, the guidance system must use a suitable steering law to convert the desired trajectory into suitable commands to the control system. This output is included in figure 3.2, where the guidance system provides the desired velocity ($\boldsymbol{\nu}_d$) and desired positions and orientations ($\boldsymbol{\eta}_d$) as the reference values to the control system. The control system will then ensure that the vessel follows the given trajectory.

Figure 3.2 shows how there are multiple ways of setting up a guidance system. It may for instance be open-loop or closed-loop, where the open-loop guidance system does not consider the craft's current states (outputted from the navigation system). A closed-loop guidance system on the other hand, uses the craft's current states to update the trajectories. This thesis will mainly focus on closed-loop guidance systems, which will be used in collision avoidance. However, the open-loop design will also be implemented for completeness, as this is used for *global* path planning.

The guidance system's task of determining a safe trajectory can be divided into the following sub-problems:

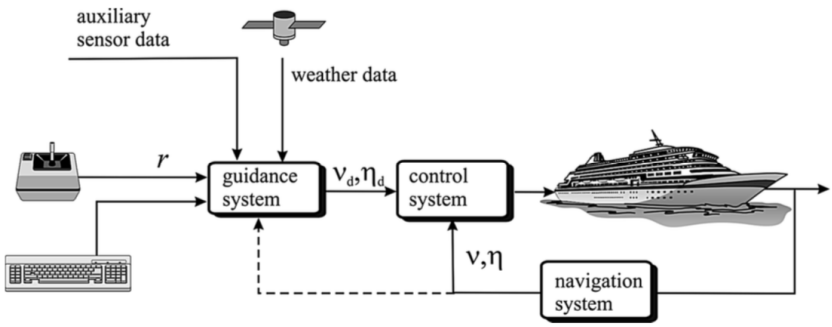


Figure 3.2: Basic outline of how the guidance system is used in a practical manner, where guidance systems can either be closed-(dotted line) or open-loops. Figure is obtained from [20, p. 242].

Step 1: Global path planning
Plan a path from given start position to an end position, where static and known obstacles are considered. This step outputs a complete sequence of states from start to end, and can be computed offline. Unlike its counterpart, (step 3) collision avoidance, which is a *local* path planner.

Step 2: Obstacle Detection
Once a complete trajectory is determined in step 1, one must be prepared for possible obstacles that were unaccounted for by the global path planner. Therefore, an obstacle detection unit is needed. This unit must continuously check surrounding environment for either static or dynamic obstacles and will call on the collision avoidance unit in the case of a newly detected obstacle.

Step 3: Collision Avoidance
Re-plan the path in a local area in order to avoid collisions with detected obstacles, the method for re-planning may vary but should be computationally effective as the re-planning is completed online. Once a new local path is generated and the obstacle is avoided, the collision avoidance unit should ensure that the path converges back to the global path from step 1.

The order in which the guidance system performs the aforementioned tasks will vary, the listing given above is merely a review of how trajectory generation typically occurs. Figure 3.3 shows how step 1-3 in the guidance system may work together. Within the global path planner and collision avoidance units, the methods used to generate the path will differ depending on the objective of the guidance system. For the case of trajectory tracking, the output of the local and global path planner will be a smooth time-varying trajectory [20]. For path following however, where the path is independent of time, a set of waypoints is generated. Either way, using the waypoints or the time-varying trajectory, the guidance system generates the necessary control signals by using steering laws. These steering laws will be

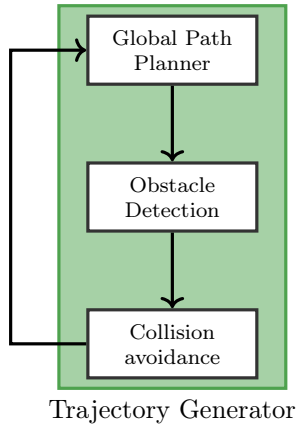


Figure 3.3: Trajectory generator within guidance system

referred to as guidance algorithms and are explained in the next section.

3.1.3 Guidance Algorithms

For the sake of this thesis, the line-of-sight (LOS) steering laws are relevant. The theory is obtained from [20], and these steering laws are used to follow straight-line paths without temporal constraints (path following).

The aim of the LOS steering law is to ensure that the craft tracks the straight lines connecting the waypoints. This is achieved by assigning a suitable desired course χ_d, in the presence of currents, or a suitable desired heading ψ_d, with no currents present. Using an autopilot to control the craft's heading towards the desired heading will then ensure proper tracking.

Now to the implementation details, the reader is referred to figure 3.4 for an illustration of the algorithm's parameters. Path-following is achieved by aligning the craft's speed vector ($\mathbf{U}$) with the LOS vector. This effectively means that the craft is heading towards the straight-line path connecting the two waypoints, $\mathbf{p}_k = [x_k,\ y_k]^T$ and $\mathbf{p}_{k+1} = [x_{k+1},\ y_{k+1}]^T$. In order to quantify the craft's current position ($\mathbf{p}$) with respect to the path between $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$, a path-fixed reference frame is used. This frame has its origin in $\mathbf{p}_k$ and is rotated at an angle α_k (see equation (3.2)) relative to the inertial frame. The position of the craft can then be written with the coordinates $\boldsymbol{\epsilon}(t) = [x_e(t),\ y_e(t)]^T$, where $x_e(t)$ is the along-track error and $y_e(t)$ is the cross-track error. Thus, the aim of the LOS steering law is to bring $\lim_{t\to\infty} y_e(t) = 0$; if temporal constraints were included then the along-track error would also have to be considered [36]. For spatial constraints only, there are two main guidance algorithms; namely *lookahead-based steering* and *enclosure-based steering*. This thesis will use the *lookahead-based steering* approach as it is valid for all values of $y_e(t)$; unlike its counterpart *enclosure-based steering*.

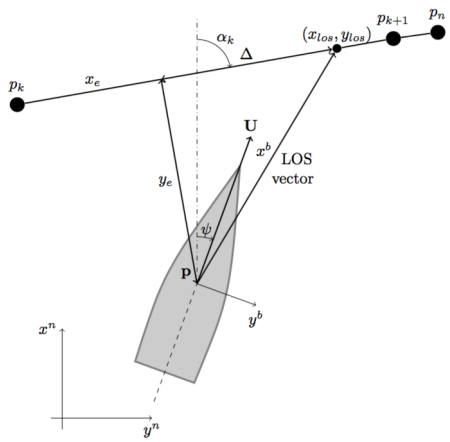


Figure 3.4: Important parameters for the LOS guidance laws. Figure is obtained from [36].

Assuming no currents, such that $\chi = \psi$, the reference heading ψ_d set by the lookahead-based steering law is:

$$\psi_d(e) = \psi_p + \psi_r(e) \tag{3.1}$$

where the two parts are given by:

$$\psi_p = \psi_k := \text{atan2}(y_{k+1} - y_k, x_{k+1} - x_k) \in [-\pi, \pi] \tag{3.2}$$

$$\psi_r(e) := \arctan\Big(\frac{-y_e}{\Delta}\Big) \in [-\pi/2, \pi/2] \tag{3.3}$$

The two course angles represent the path-tangential angle and the velocity-path relative angle respectively. Within ψ_r, y_e is the current cross-track error and Δ is the lookahead distance. Considering the projection of the craft's position in NED ($\boldsymbol{p}$) onto the path-fixed reference frame $\boldsymbol{p}_p$, Δ is the distance ahead of $\boldsymbol{p}_p$ that the craft's velocity vector will be directed towards by using ψ_d from equation (3.1) as the reference value to the autopilot. See figure 3.4 for an illustrative explanation.

Once the lookahead-based algorithm is implemented, a switching mechanism is needed. This mechanism switches the steering law's attention from one section of the path to the next; meaning that its focus switches to the straight line between the next waypoints. If $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k+1}$ are currently considered, then the switching mechanism will switch to consider $\boldsymbol{p}_{k+1}$ and $\boldsymbol{p}_{k+2}$. According to [20, p. 264], a circle of acceptance R_{k+1} can be used to check if the steering law should switch to the next segment of the path. If the craft's position satisfies equation (3.4), then the switching is initiated and the craft starts to move towards the next waypoint.

$$(x_{k+1} - x(t))^2 + (y_{k+1} - y(t))^2 \leq R_{k+1} \tag{3.4}$$

3.2 COLREGS - The Marine Rules of the Road

The COLREGS are the International Regulations for Preventing Collisions at Sea, and presents a comprehensive rule book for traffic schemes at sea. The rules were introduced in 1972, and were a result of a revised version of the Collision Regulations from 1960 [15].

This section investigates Part B of COLREGS, which covers steering and sailing rules. It is assumed throughout this thesis that the all vessels, both the own vessel and the target vessels, are power-driven vessels. This means that all vessels are driven by some sort of machinery, sailing vessels are therefore not included. Further, the quantification of COLREGS in a collision scenario is also covered. This part is important for the collision avoidance algorithm outlined in section 4.3.

3.2.1 Steering and Sailing Rules

The most relevant parts of the steering and sailing rules are presented, as these will make up the scenarios used to test the developed collision avoidance system. The interested reader is referred to [25] for further details regarding COLREGS. Note that all rules are collected from the source in [25], additionally rules 13-15 are illustrated in figure 3.5.

Rule 8: Action to avoid collision

(a). *Any action to avoid collision shall be taken in accordance with the rules, if the circumstances of the case admit, be positive, made in ample time and with due regard to the observance of good seamanship.*

(b). *Any alteration of course and/or speed to avoid collision shall, if the circumstances of the case admit, be large enough to be readily apparent to another vessel observing visually or by radar; a succession of small alterations of course and/or speed should be avoided.*

Rule 13: Overtaking situation

(a). *Notwithstanding anything contained in the Rules of part B, sections I and II, any vessel overtaking any other shall keep out of the way of the vessel being overtaken.*

(b). *A vessel shall be deemed to be overtaking when coming up with another vessel from a direction more than 22.5 degrees abaft her beam, that is, in such a position with reference to the vessel she is overtaking, that at night she would be able to see only the sternlight of that vessel but neither of her sidelights.*

Rule 14: Head-on situation

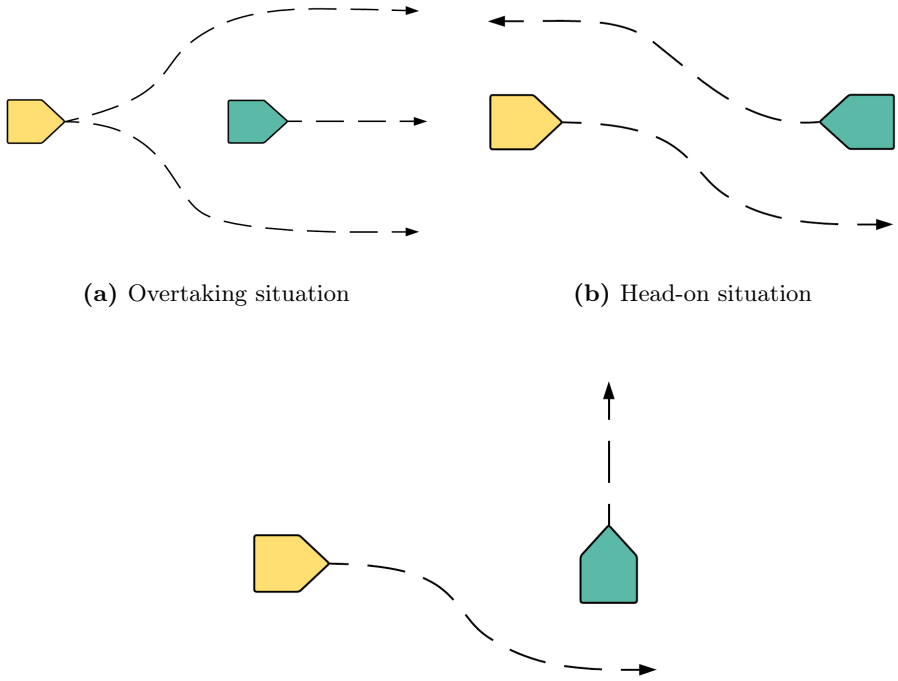
When two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other.

Rule 15: Action by give-way vessel

Every vessel which is directed to keep out of the way of another vessel shall, so far as possible, take early and substantial action to keep well clear.

Rule 16: Crossing situation

When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel.



(a) Overtaking situation

(b) Head-on situation

(c) Crossing situation

Figure 3.5: Collision avoidance routes adhering to COLREGS, where the yellow ASV is the give-way vessel and the teal ASV is the stand-by vessel.

3.2.2 Quantifying COLREGS

If the aforementioned COLREGS rules are to be interpreted by software, each collision situation must be categorized by some numerical means. Following the procedure from [51], the relative bearing (β) is used to quantify COLREGS. The value of β decides which COLREGS rules apply to a given collision scenario. However, before the COLREGS are applied, the software has to determine *if* there is risk for a collision. This is completed using the closest point of approach (CPA) method. Both definitions for β and CPA are presented below.

Closest Point of Approach (CPA)

The closest point of approach (CPA) uses the current position and velocity of the ASV and target vessels to determine if there is a risk for a collision. The details of the CPA method are obtained from [31], and are as follows.

$$t_{cpa} = \begin{cases} 0 & \text{, if} \|v_A - v_B\| \leq \epsilon \\ \frac{(p_A - p_B) \cdot (v_A - v_B)}{\|v_A - v_B\|^2} & \text{, otherwise} \end{cases} \tag{3.5}$$

$$d_{cpa} = \big\|(p_A + v_A t_{cpa}) - (p_B + v_B t_{cpa})\big\| \tag{3.6}$$

The t_{cpa} value is the time until the vessels reach their closes point of approach, whereas d_{cpa} is the distance between them at this point. In order to signal a risk of an imminent collision, the following criteria for t_{cpa} and d_{cpa} must be satisfied simultaneously.

$$0 \leq t_{cpa} \leq t_{max} \tag{3.7}$$

$$d_{cpa} \leq d_{min} \tag{3.8}$$

Relative bearing

The relative bearing (β) represents the clockwise angle between the heading of the vessel, which in this case is the target vessel, and a straight line drawn from the target vessel onto the approaching vessel, which is the own vessel.

The value for β is calculated through equations (3.9)-(3.12), where "TV" refers to the target vessel and "OV" refers to the own vessel. Equation (3.9) is the line-of-sight vector between the two vessels, equation (3.10) is a vectorial representation of the North direction, equation (3.11) is the true bearing of the obstacle (i.e the target vessel); and finally, equation (3.12) is the relative bearing between the target vessel and the own vessel.

$$LOS = \underbrace{\begin{bmatrix} N_{TV} \\ E_{TV} \end{bmatrix}}_{p_{TV}} - \underbrace{\begin{bmatrix} N_{OV} \\ E_{OV} \end{bmatrix}}_{p_{OV}} \tag{3.9}$$

$$NORTH = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3.10}$$

$$\beta_{TB} = \arccos\left(\frac{LOS \cdot NORTH}{\|NORTH\| \times \|LOS\|}\right) \tag{3.11}$$

$$\beta = \beta_{TB} - \psi_{TV} \tag{3.12}$$

Once β is calculated, the applicable COLREGS situation is determined by the means of figure 3.6, where the zones for β are based on the procedure in [51]. From this figure, the COLREGS rules are divided by the following values for β:

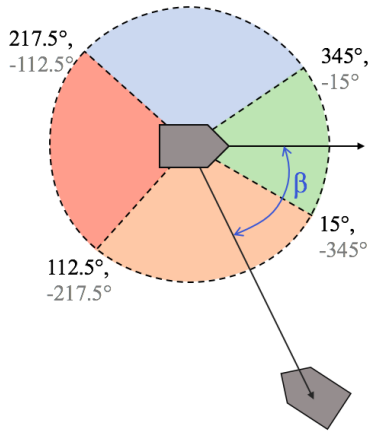


Figure 3.6: The figure shows how the relative bearing, β, is used to classify which COLREGS rules apply to a given collision situation. The green area identifies a head-on situation, the orange area is crossing from the right, the red area represents an overtaking situation, and finally, the blue area is crossing from the left.

Head-on:
$\beta \in [-15°, 15°] \cup ([345°, 0°] \cap [0°, 15°]) \cup ([-345°, 0°] \cap [0°, -15°])$

Crossing from the right:
$\beta \in [15°, 112.5°] \cup [-345°, -217.5°]$

Overtaking:
$\beta \in [112.5°, 217.5°] \cup [-217.5°, -112.5°]$

Crossing from the left:
$\beta \in [217.5°, 345°] \cup [-112.5°, -15°]$

3.3 Modelling of Surface Vessels

Surface vessel modelling is needed to comprehend the motions of ASVs, which is crucial in the generation of a feasible path. This section merely includes the basic theory of these models, as [20] covers the modelling in great detail. The interested reader is therefore referred to part one of [20] for more background theory.

3.3.1 Kinematic Model

The kinematic model considers only the geometrical aspects of the vessel's motion [20, p. 15]. This thesis will use a 3 DOF vessel model, where the horizontal motion of the vessel is studied. The North-East-Down (NED) reference frame is the inertial

frame, whereas the moving reference frame that is fixed to the vessel is referred to as the BODY reference frame. This frame is used to consider how the velocities ($\boldsymbol{\nu}$) affect the positions ($\boldsymbol{\eta}$) in NED. The kinematic relationship between the BODY and the NED frame is described by equation (3.13), where the Euler angles are used to transform the BODY-fixed velocity vector into the NED frame. Note that the only Euler angle needed for a 3 DOF surface vessel is the yaw angle (ψ), also referred to as the vessel's heading in subsequent chapters.

$$\underbrace{\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{\psi} \end{bmatrix}}_{\dot{\boldsymbol{\eta}}} = \underbrace{\begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{R}(\psi)} \underbrace{\begin{bmatrix} u \\ v \\ r \end{bmatrix}}_{\boldsymbol{\nu}} \tag{3.13}$$

3.3.2 Kinetic Model

The kinetic model studies how the forces on the model affect its motion. There are numerous representations for this model, all with varying degrees of accuracy. The most general equation of motion is presented in equation (3.14), from [20, p. 15]. Note that $\boldsymbol{\tau}$ refers to actuator forces, and $\boldsymbol{\tau}_{wind}$ and $\boldsymbol{\tau}_{wave}$ are environmental forces. The left-hand side of the equation on the other hand, refers to hydrostatic and hydrodynamic forces that affect the vessel.

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta}) + \mathbf{g}_0 = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave} \tag{3.14}$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + C_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + C_A(\boldsymbol{\nu})\boldsymbol{\nu} + D(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave} \tag{3.15}$$

Equation (3.15), from [20, p. 134], shows the kinetic model for a 3 DOF horizontal plane model. This model is used to developed the implemented ASV model in section 5.2.1. The subscripts "RB" and "A" refer to the *rigid-body* and the *added-mass* kinetics respectively.

3.4 Modelling of Stochastic Processes

A deterministic system is completely predictable, whereas the outcome of a stochastic system is uncertain. Karatzas and Shreve define a *stochastic process* as a mathematical model that portrays a random phenomenon [28], whereas Coleman describes it as a system that evolves over time while undergoing chance fluctuations [13]. Nevertheless, central to the theory of a stochastic process, is the *random variable*.

Definition 3.1. A random variable, referred to by an upper case letter X, is a function that associates a numerical value, referred to by a lower case letter x, with each possible outcome of the stochastic process [55]. □

Using definition 3.1, a stochastic process can be described by a set of random variables as shown in equation (3.16). In this case, t refers to the time at which the numerical value of the random variable X is collected and can take any value in the subset of $\{-\infty, \infty\}$. The values the random variable X can take are called its *states*, and a change in X from one value to another is called a *transition* between the states [13]. Further explanations regarding stochastic processes will assume that the reader has basic knowledge of statistical measures, such as mean and variance.

$$X(t) = \begin{bmatrix} X(t_1) \\ X(t_2) \\ \vdots \\ X(t_n) \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \tag{3.16}$$

3.4.1 Probabilistic Description of Stochastic Processes

In order to characterize various stochastic processes, one must use statistical properties. Probabilistic terms such as *probability density functions*, *autocorrelation* and *power spectral density functions* will be explained as they are of relevance for the topics to be covered in this thesis.

Probability Density Function

Definition 3.2. A probability density function (PDF) $f(x)$, often referred to as a density function, for a continuous random variable X has the following properties [55]

1. $f(x) \geq 0$, for $\forall x \in R$
2. $\int_{-\infty}^{\infty} f(x)dx = 1$
3. $P(a < X < b) = \int_a^b f(x)dx$

□

Based on definition 3.2, a PDF is related to a stochastic process in that it describes the likelihood of obtaining a numerical value x for a random variable X at each time instance t.

For stochastic processes, which is a series of random variables, it is possible that each random variable X_i in equation (3.16), has a PDF unlike the other random variables constituting the process [8]. Thus, the probability distribution for X_1 may differ from X_2 even though they make up the same stochastic process. Further, the notion of a *stationary* stochastic processes refers to a process where its density functions remain unchanged by a translation in time. This means that the density functions of the set of random variables given by the stochastic process $X(t) = \{X(t_1), ..., X(t_n)\}$ will be equal to the translated set given by $X'(t) = \{X'(t_1 + \tau), ..., X'(t_n + \tau)\}$ [8].

Autocorrelation Function

Definition 3.3. The autocorrelation of a stochastic process is used to describe how a process is correlated to itself at two different sampling times. The notation and formal definition is given by:

$$R_X(t_1, t_2) = E[X(t_1)X(t_2)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 x_2 f_{x_1 x_2}(x_1, x_2)dx_1 dx_2 \tag{3.17}$$

$$R_X(\tau) = E[X(t)X(t+\tau)] \tag{3.18}$$

where E[...] refers to the expected value, $X(t_i)$ is the random variable of the stochastic process at time t_i, and equation (3.18) refers to a *stationary* stochastic process [8].

□

The autocorrelation function described in equation 3.17 calculates the expected value of the product of X_1 and X_2, which in turn gives the correlation between the two random variables. In statistics, the term *correlation* is used to refer to the numerical measure of the strength of a relationship between random variables. Thus, a larger value for an autocorrelation at two time instances will imply that the stochastic process has a stronger relationship with itself at various times. For the stationary case in equation (3.18), if the autocorrelation function decreases once τ decreases, it means that the stochastic process will change rapidly with time as there is little correlation between two values taken in close proximity with regard to time.

Power Spectral Density Function

Definition 3.4. The power spectral density function, also called the spectral density function, for a stationary process is formally defined as:

$$S_X(j\omega) = \mathcal{F}[R_X(\tau)] = \int_{-\infty}^{\infty} R_X(\tau)e^{-j\omega\tau}d\tau \tag{3.19}$$

where $\mathcal{F}$ refers to the Fourier transform of the autocorrelation [8]. □

From definition 3.4, it is clear that the autocorrelation and spectral density function are closely related. As they are Fourier transform pairs, they are equivalent. However, the spectral density function relates to the frequency content of the stochastic process $X(t)$ whereas the autocorrelation looks at the correlation between two samples. That is, the amplitude of the spectral density function will portray the power contained in the process $X(t)$ at each specified frequency. The relation between the autocorrelation and spectral density function will be made clearer by investigating how they are related to the mathematical abstraction of white noise in section 3.4.3.

3.4.2 Normal Distribution

The normal distribution, also called a Gaussian distribution, is of great importance in modelling natural phenomenon. Its importance is related to the Central Limit Theorem, explained in definition 3.5, which ensures that it frequently appears in nature. The most relevant properties of the normal distribution, for the sake of this thesis, are listed in definition 3.6.

Definition 3.5. According to the *Central Limit Theorem*, the probability density distribution of a set of sample means will approach a normal distribution as the sample size grows. This fact will hold no matter the shape of the population distribution, i.e the probability density distribution of the population the samples originate from. For sample sizes over 30, the Central Limit Theorem is especially accurate.

□

Definition 3.6. The normal distribution has the following properties:

1. The probability density function is symmetric about the mean, and is completely defined once the mean μ and the standard deviation σ are set.
2. For a random variable X, the normal distribution has a probability density function $n(x;\mu,\sigma)$, plotted in figure 3.7. $n(x;\mu,\sigma)$ follows the general properties of probability density functions, as listed in definition 3.2, and is given by equation (3.20).

$$n(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\Big(\frac{-(x-\mu)^2}{2\sigma^2}\Big) \tag{3.20}$$

□

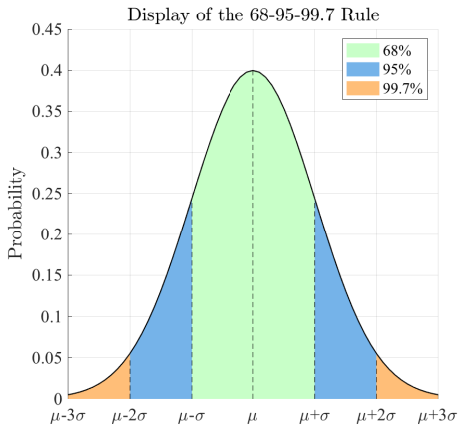


Figure 3.7: The probability density function of a normal distribution, highlighting how the 68-95-99.7 rule relates to the distribution's mean μ and standard deviation σ.

3.4.3 White Noise

A widely used stochastic process in the modelling of uncertainty is *white noise*. It is a mathematical abstraction that is useful as many real processes can be approximated as white noise. In statistics, the term white noise is used to describe a stochastic process with a constant spectral density. In physics on the other hand, white noise refers to a signal that contains all frequencies with equal intensity. These two descriptions of white noise may differ, but using definition 3.4, it is clear that they are equivalent.

The statistical properties of white noise; mean, variance and covariance; are given in equations (3.21)-(3.23) respectively. Here $F(t)$ is used to refer to a white noise process.

$$E[F(t)] = 0 \quad \forall t \tag{3.21}$$

$$Var[F(t)] = \sigma^2 \quad \forall t \tag{3.22}$$

$$Cov[F(t), F(s)] = 0 \quad \forall t \neq s \tag{3.23}$$

As white noise is a stationary stochastic process, its autocorrelation function is given by equation (3.18). Using the statistical interpretation of white noise, equations (3.24) and (3.25) describe the spectral density and autocorrelation functions respectively. The delta dirac function in equation (3.25) means that white noise has an infinite autocorrelation at $\tau = 0$, and zero autocorrelation at $\tau \neq 0$. In practice, this means that two samples of white noise at two different time instance will be completely unrelated to each other.

$$S_{wn}(jw) = \sigma^2 \tag{3.24}$$

$$R_{wn}(\tau) = \sigma^2\delta(\tau) \tag{3.25}$$

3.4.4 Wiener Process

The Wiener process, also known as Brownian motion, can be used to describe an integrator driven by white noise. For this reason, the wiener process is often used in the solution of stochastic differential equations where an uncertainty modelled by white noise is present. In order to properly explain the Wiener process, it is necessary to define the *Markov* property.

Definition 3.7. A discrete-time stochastic process, $(X_n)_{n\in N}$, has the Markov property if, for all $n \geq 1$, the probability distribution of X_{n+1} is solely determined by the state of the process at time n, and is independent of previous states at times $k \leq n-1$ [46]. Written in terms of probabilities, the Markov property is as follows:

$$P(X_{n+1} = x_{n+1}|X_n = x_n, ..., X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n) \tag{3.26}$$

□

The Wiener process stems from the notion of a random walk, where one starts at the origin and takes a step either forward or backward, with an equal probability of stepping in either direction. This means that the position after n steps will be random and the direction of the walk will only depend upon the current direction of the walk. Using definition 3.7, it is then clear that the Wiener process has the Markov property.

The Wiener process is defined as the continuous analog of the discrete random walk [8]; that is, the wiener process is the resulting path obtained by integrating white noise over a time interval. Using $F(t)$ to represent a white noise process, the Wiener process $(W(t))$ is then as follows:

$$W(t) = \int_0^t F(u)du \tag{3.27}$$

By using the statistical properties of white noise from equations (3.21)-(3.23), the definition of the stationary autocorrelation function in equation (3.18), and the mean-square value of the Wiener process in equation (3.28); the mean and variance of the Wiener process are given by equations (3.29) and (3.30) respectively [8].

$$E[W(t)^2] = E\Big[\int_0^t F(u)du \int_0^t F(v)dv\Big] = \int_0^t \int_0^t E[F(u)F(v)]dudv$$
$$= \int_0^t \int_0^t R_F(u-v)dudv = \int_0^t \int_0^t \sigma^2\delta(u-v)dudv = \sigma^2 \int_0^t dv = \sigma^2 t \quad (3.28)$$

$$E[W(t)] = E\Big[\int_0^t F(u)du\Big] = \int_0^t E[F(u)]du = 0 \quad (3.29)$$
$$Var[W(t)] = E[W(t)^2] - E[W(t)]^2 = E[W(t)^2] = \sigma^2 t \quad (3.30)$$

As the variance of the Wiener process in equation (3.30) depends upon the time (t), it will change continuously; making the Wiener process a non-stationary stochastic process. The previously listed statistical properties of the Wiener process have been used to create the following definition of the process.

Definition 3.8. The Wiener process ($W(t)$) is defined over the continuous interval $t \in [0, T]$ and satisfies the following conditions [24].

1. $W(0) = 0$
2. $W(t) - W(s) \sim \sqrt{t-s}N(0,1)$, where $0 \leq s < t \leq T$ and N(0,1) denotes a normal distribution with zero mean and unity standard deviation
3. The two increments $W(t) - W(s)$ and $W(v) - W(u)$ are independent, where $0 \leq s < t < u < v \leq T$

□

3.4.5 Stochastic Calculus

Stochastic calculus applies mathematical theories, such as integral and differential calculus, on random processes. This enables us to study how a stochastic process is set to change over time; a study that is of importance for this thesis.

Stochastic Differential Equations

When a stochastic process changes with time and is modelled mathematically, the resulting equation is a stochastic differential equation (SDE). Contrasting to an ordinary differential equation (ODE) and its solution, in equations (3.31) and (3.32), the solution to a stochastic differential equation, in equations (3.33) and (3.34), appears to be quite similar.

$$\frac{dx}{dt} = f(x), \quad x(t_0) = x_0 \tag{3.31}$$

$$x(t) = x_0 + \int_{t_0}^{t} f(s)ds \tag{3.32}$$

In the SDE below, both $X(t)$ and $Y(t)$ are stochastic processes and X_0 is a random variable.

$$dX(t) = f(X(t))dt + g(X(t))dY(t), \quad X_0(t_0) = x_o \tag{3.33}$$

$$X(t) = X_0 + \underbrace{\int_0^t f(X(s))ds}_{\text{Deterministic integral}} + \underbrace{\int_0^t g(X(s))dY(s)}_{\text{Stochastic Integral}} \tag{3.34}$$

Despite the similarities between the solutions for $x(t)$ and $X(t)$, their behaviours differ. First of all, the solution of the deterministic differential equation will remain unchanged from each time it is solved; the stochastic solution on the other hand will differ from time to time. This difference is rooted in the random variables present in the solution for $X(t)$. Further, the solution of the stochastic integral in equation (3.34) must be solved differently than that of the deterministic integral in equation (3.32). This will be explained further in the next sections.

Stochastic Integrals

A deterministic integral on the form in equation (3.35) can be approximately solved by using the Riemann sum over the desired interval. There are two alternative forms to the Riemann sum, given by equations (3.36) and (3.37). The exact solution to the integral is obtained by allowing $\Delta t = t_{i+1} - t_i \to 0$ for the Riemann sum, where both alternatives to the Riemann sum will produce the same, and correct, and answer.

$$I = \int_0^T h(t)dt \tag{3.35}$$

$$I \approx \sum_{i=0}^{N-1} h(t_i)(t_{i+1} - t_i) \tag{3.36}$$

$$I \approx \sum_{i=0}^{N-1} h\Big(\frac{t_i + t_{i+1}}{2}\Big)(t_{i+1} - t_i) \tag{3.37}$$

This way of defining solutions to integrals is used in an equivalent manner for stochastic integrals. The stochastic integral from equation (3.34) is expressed in

equation (3.38), where $g(X(s))$ is replaced with $h(t)$ to allow for consistency to the notation in equation (3.35). In order to approximate a solution to this integral, it is possible to use two different forms of the Riemann sum; much like the case for deterministic integrals. Here equations (3.39) and (3.40) correspond to equations (3.36) and (3.37) respectively. Recall that $Y(t)$ is, in this case, a stochastic process.

$$I = \int_0^T h(t)dY(s) \tag{3.38}$$

$$I \approx \sum_{i=0}^{N-1} h(t_i)(Y(t_{i+1}) - Y(t_i)) \tag{3.39}$$

$$I \approx \sum_{i=0}^{N-1} h\Big(\frac{t_i + t_{i+1}}{2}\Big)(Y(t_{i+1}) - Y(t_i)) \tag{3.40}$$

Unlike the deterministic case, the two approximations to the stochastic integral will give rise to different answers as $\Delta t = t_{i+1} - t_i \to 0$ [24]. Using the sum in equation (3.39) to approximate a stochastic integral gives rise to the *Itô integral*, and the sum in equation (3.40) gives rise to the *Stratonovich integral*.

The Euler-Maruyama Method

Much like the Euler method is used to numerically solve deterministic ordinary differential equations (ODEs), the *Euler-Maruyama* method is used to numerically solve stochastic differential equations (SDEs). It is assumed that the reader has sufficient knowledge with numerics, hence, only the extension of numerics into stochastic calculus will be covered in this section.

The Euler-Maruyama method can be used to solve a scalar, autonomous SDE; a kind of SDE that is shown by equation (3.34). Here the term *autonomous* refers to the mathematical sense of the word, meaning that an autonomous differential equation is one that does not specifically depend on the independent variable. For a differential equation used to portray a physical system, the independent variable is typically time. Thus, a differential equation where time is not an explicit variable, in the sense that the states portrayed in the differential equation change with time but time is not present as a variable, is an autonomous differential equation.

According to [24], SDEs are generally portrayed in their differential form rather than in their integral form, as shown by equations (3.33) and (3.34) respectively. Therefore, the Euler-Maruyama method will be presented using the differential form of the SDE.

As with any numerical method, the first step in solving the SDE is to discretize the interval of interest. Working with mathematical models of physical systems, this means that the time interval is discretized into N parts. Much like the Wiener

process was defined over a specified time interval, the solution of the SDE will also be over a time interval given by $t \in [0, T]$. Using that $\Delta t = T/N$, $t_i = i\Delta t$ and $i \in [0, N]$; the Euler-Maruyama solution of the SDE in equation (3.33) is given by equation (3.41). Note that both $X(t)$ and $Y(t)$ represent stochastic processes.

$$X_{i+1} = X_i + f(X_i)\Delta t + g(X_i)(Y(t_{i+1}) - Y(t_i)) \tag{3.41}$$

In order to determine X_{i+1} using equation (3.41), the value of $Y(t_{i+1}) - Y(t_i)$ must be set. This value will depend on the type of stochastic process that $Y(t)$ represents. As this thesis will always use the Wiener process $W(t)$ as $Y(t)$, the value that needs to be determined will be $W(t_{i+1}) - W(t_i)$. As explained in section 3.4.4, the random variable representing an increment in the Wiener process follows a normal distribution with zero mean and a a variance equal to size of the increment. For instance, if the Wiener process is discretized with an increment equal to $\Delta t = t_{i+1} - t_i$, then the value of the random variable $W(t_{i+1}) - W(t_i)$ can be sampled from a normal distribution $\sim \Delta t N(0, 1)$.

3.5 Network Representation

This section presents the basic concepts of network representation, and is provided to shed some light on the terminology used to describe the A* search in section 3.6. The terminology used in this thesis is collected from [38].

Motion planning is an optimization problem, and its nature allows it to be presented as a *network* that consists of *nodes* connected by *arcs*. The nodes will typically represent positions in space, whereas the arcs can represent various aspects of the motion planning problem. They may for instance represent the cost of moving between two nodes, the distance between two nodes, the control input needed to move the vehicle from one node to the next, and so on. The specific representation used for this thesis will be covered in section 5.3.

Figure 3.8 shows how a simple graph can be presented using nodes and arcs. This graph G = (N,A) consists of a set of nodes N = {A, B, ..., G} and arcs A = {a, b, ..., j}, and is referred to as a network only when specific numerical values are associated to the nodes and their arcs. The arcs in figure 3.8 are *directed*, meaning that they only allow flow in the direction specified by the arrow. An *undirected* graph on the other hand, allows flow in both directions along an arc.

3.6 A* search

The A* search is a popular computer algorithm for path finding and graph traversals. Using a predefined graph, the algorithm finds the optimal set of nodes that leads to the target node by introducing a *heuristic*. The heuristic represents an

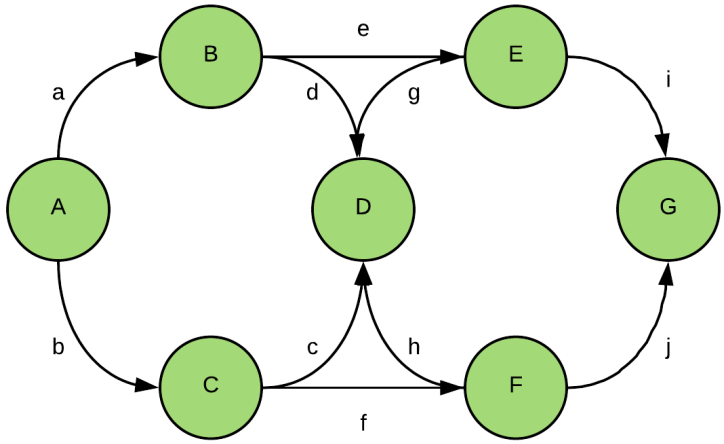


Figure 3.8: Basic network representation using nodes, shaded in green, and arcs, labelled from a to j.

estimated cost of moving from a node towards the target node, and is included to provide the A* search with some brains. The A* algorithm can be applied to a wide variety of problems, but will be explained in the motion planning context as that is the topic of this thesis.

Consider the problem of moving from start to goal in figure 3.9, figure 3.9a is able to detect the obstacle before getting trapped, unlike the path finding algorithm shown in figure 3.9b. The A* algorithm is typically of the kind in figure 3.9a, in that a properly designed heuristic will be able to detect future traps. A simple example is presented below to highlight the basic concepts of the A* algorithm.

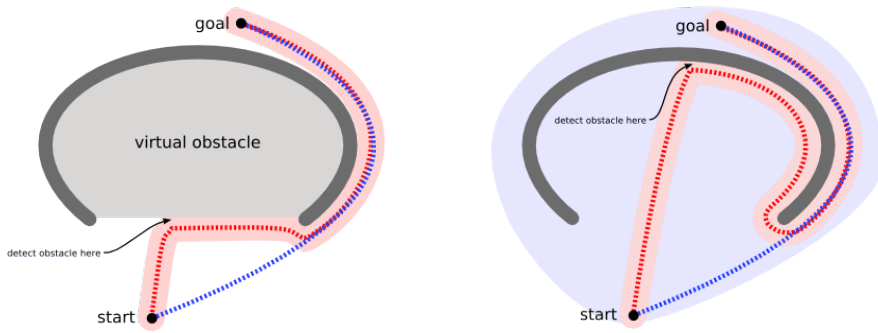


Figure 3.9: The two figures show an algorithm that uses brains to determine the optimal path in (a) and one that does not use brains in (b). Images collected from [26].

3.6.1 Introduction to A* search

Before using the A* search for path finding, a network has to be created. For motion planning for ASVs, the network should represent the environment the search is to be completed over. Once the network is created and the starting node and target node are specified, the A* search can start. The goal of the algorithm is to determine the lowest-cost path from start to target.

The cost of a node, say node $n = E$ in figure 3.8, is determined by a cost-to-go term $h(n)$ and a cost-to-come term $g(n)$. The cost-to-come term is the cost to go from the starting node, node $n = A$ in figure 3.8, to the current node $n = E$. For this example, the cost to go from node A to E would be $g(E) = a + e$. The cost-to-go is the heuristic, as it represents an estimate of the cost to go from the current node $n = E$ to the target node $n = G$. For this example, the cost to go from node E to G would be $h(E) = i$. The total cost of node E is then equal to the sum of $g(E)$ and $h(E)$, $f(E) = g(E) + h(E)$. The total cost for a generic node n in a network in an A* search is then given by:

$$f(n) = g(n) + h(n) \tag{3.42}$$

The A* algorithm uses the total cost in equation (3.42) to expand the node with the lowest cost. With a properly designed heuristic, this can lower the amount of expanded nodes such that the time-complexity of the search is greatly reduced. Using figure 3.8 as an example, if the current node is $n = B$, then the path can either go to node $n = D$ or $n = E$. The A* search chooses the next node, either D or E, with the lowest total cost $f(n)$. If node E has the lowest cost, and node G is the target node, then the A* search will give the optimal path from A to G as: {A, B, E, G}. As only the lowest-cost nodes are expanded, the search will then only have opened nodes: {A, B, D, E, G}; effectively leaving out nodes {C, F} and

thereby saving the amount of memory needed to complete the search and reducing the time spent to find the optimal path.

The cost-to-come function $g(n)$ is, as exemplified above, typically the length from the start node n_s to the current node n. This length can be calculated in numerous ways, but will be calculated as the euclidean distance in this thesis. Making the cost-to-come, for a node n with position (x,y) in a 2D-plane, equal to:

$$g(n) = \sqrt{(x - x_s)^2 + (y - y_s)^2} \tag{3.43}$$

where (x_s, y_s) represents the coordinates of the start node.

The design of a heuristic $h(n)$, also called cost-to-go, is more complex. It is also a key factor in the implementation of an A* search and is vital for the results of this thesis. Therefore, it is explained in a separate section below.

3.6.2 Heuristics

The time complexity of the A* search is influenced by the design of the heuristic. Note that the heuristic is an estimate of the *minimum* cost to go from a node n to the target node n_t. If the designed heuristic is length, then the heuristic of a node n would be the minimum length to go from n to n_t.

As the heuristic is an estimate, it can either underestimate, overestimate or perfectly estimate the true cost-to-go [23]. The effect of each these cases on the A* search is as follows:

- If the heuristic underestimates the true cost, then A* has to expand more nodes to reach the target, effectively making it slower. However, it is guaranteed to find the shortest path.
- If the heuristic overestimates the true cost, then the A* search will be faster but it is not guaranteed to find the shortest path towards the target node.
- A perfect estimate of the heuristic means that the algorithm only expands the nodes that lead to the shortest-path, resulting in a fast algorithm and an accurate result. Although, obtaining a perfect estimate of the cost-to-go is difficult to complete in practice.

A commonly used design for the heuristic function $h(n)$ is the length to go from the current node n to the target node n. Much like the distance calculation of the cost-to-come $g(n)$, this distance can be calculated in numerous ways. If the euclidean length is used, $h(n)$ would be equal to $g(n)$ in equation (3.43), where (x_s, y_s) is then replaced by the coordinates of the target node (x_t, y_t). This thesis will use a different method to give an estimate of the cost-to-go, the explanation of the implemented heuristic function is presented in sections 4.2.1 and 4.3.1.

3.6.3 Pseudocode

An implementation of the A* search in any programming language requires the system to keep track of two lists:

1. **OPEN:**
   Contains the nodes that have been opened throughout the search. A node is opened once its parent is chosen as the current node, which means that it has been the node with the lowest total cost ($f(n)$) in the OPEN list.
2. **CLOSED:**
   Contains the nodes that have been chosen as the current node due to their minimum total cost value ($f(n)$).

The configuration of the OPEN list should contain the current node, its parent node and its costs. It is important to keep track of the parent nodes as the algorithm backtracks through the parent nodes once the target node is opened to find the optimal path. An example of the set-up of an OPEN list is given in table 3.1, where the first column is a boolean value used to indicate if the node is still in OPEN (open = 1) or in CLOSED (open = 0). The CLOSED list on the other hand, requires less information. It merely keeps track of the nodes that have already been explored, and should only contain these nodes. An example of how a CLOSED list may look like is shown in table 3.2.

Table 3.1: Example of an OPEN list

Open	Node n	Parent Node n_p	$g(n)$	$h(n)$	$f(n)$
1/0	⋮	⋮	⋮	⋮	⋮

Table 3.2: Example of a CLOSED list

Node n
⋮

Once the set-up of the OPEN and CLOSED lists is defined, the coding of the search can start. The implementation of the A* search can be represented by the pseudocode in algorithm 1, presented with a MATLAB-like syntax.

Algorithm 1: A* search

```
% Complete the search
Initialize OPEN to contain the start node
Initialize CLOSED to an empty list
while target node is not in OPEN do
    find node with lowest f(n) value in OPEN
    make said node the current node
    if current node is target node then
        end search
    else
        place current node in CLOSED and open its successor nodes
        for each of the successor nodes do
            if successor is in OPEN and it has a lower f(n) value than that
              already listed in OPEN then
                replace the successor's f(n) value with the new, lower f(n)
                  value
                replace the parent node of the successor in OPEN to the
                  current node
            else if successor is in CLOSED and it has a lower f(n) value
              than that already listed in OPEN then
                replace the successor's f(n) value with the new, lower f(n)
                  value
                replace the parent node of the successor in OPEN to the
                  current node
            else if successor is not in OPEN or CLOSED then
                add successor to OPEN
        end
    end
end
% Backtrack from target node to start node to find optimal path
Initialize OPTIMAL to contain the target node
Make current node equal to the target node
while current node is not equal to start node do
    find parent node of current node in OPEN
    place parent node at the back of OPTIMAL
    make current node equal to the parent node
end
```

Chapter 4

Motion Planning using Path-of-Probability

In order to properly distinguish the POP algorithm from the other theory and background from chapter 3, this chapter is presented. Its aim is to clarify how probability theory and an A* search can be combined to give a motion planner with POP. First however, the general outline of the POP algorithm is presented in a step-by-step manner. Then, its fusion with the A* search for global path planning is outlined, closely followed by its extension into a dynamic environment. Note that all implementation details regarding the POP algorithm are made clear in section 5.3.3.

As mentioned in chapter 2, the POP algorithm has been used for a variety of objectives, ranging from motion planning for a flexible needle to a rolling robot. As this thesis will regard motion planning for an ASV, which draws more similarities to a rolling robot than a flexible needle, the description of the algorithm will follow the set-up for a rolling robot; as described by Wooram Park and Jaeyon Lee in [34].

4.1 The POP Algorithm

The POP algorithm from [34] is adapted to an ASV and explained below. Section 4.2 enhances the original POP algorithm by combining it with an A* search approach for global path planning, which is further used in collision avoidance according to section 4.3.

The general methodology of the POP algorithm for path planning is summarized by the steps given below. A more thorough explanation of each step follows.

Step 1: Create the SDE
Identify a suitable stochastic differential equation (SDE) that can be used to portray the state transitions of the system at hand. In path planning for ASVs, the states will be the position in the NED-plane.

Step 2: Discretize inputs
Discretize the inputs that are used in the SDE to alter the states. For an ASV, the inputs can range from the actuator commands to the velocities, where the chosen input depends on the SDE from step 1.

Step 3: Determine the candidate points
Using the discretized inputs from step 2, determine a set of candidate points for the next segment of the path.

Step 4: Solve the SDE
For each candidate point, solve the SDE a number of times (N_{SDE}) to obtain a set of trajectories that move towards the target position. Remove the trajectories that pass through or too close to a static obstacle.

Step 5: Build the PDF
Use the end-points from each trajectory generated in step 4 to estimate a probability density function (PDF) for each candidate point. Each candidate point will then have its respective PDF.

Step 6: Choose the optimal candidate point
Using the PDFs from step 5, the probability of reaching the target position from each candidate point is calculated. The candidate point with the largest probability of reaching the final position is chosen as the next waypoint in the guidance system.

Step 7: Check the termination criteria
If the candidate point chosen as the next waypoint is sufficiently close to the final position, terminate the algorithm. If not, return to step 3 and repeat the process.

The theory regarding the SDE and its solution, mentioned in step 1 and 4, is covered in section 3.4.5. Further, the creation of the SDE for an ASV, step 1, is covered in chapter 5. Therefore, these steps are not explained in detail in this section. Note that step 4, solving the SDE, can be completed in numerous manners. In this thesis however, the Euler-Maruyama method from section 3.4.5 is used, and therefore used in the description of the POP algorithm.

The discretization of the inputs in step 2 is required to obtain a finite set of candidate points for the upcoming section of the path. This discretization will depend upon the system at hand, where its constraints and feasible motions will have to be accounted for. This step can be completed by using a deterministic differential equation of the system, and simulating the effect of each input on the system by

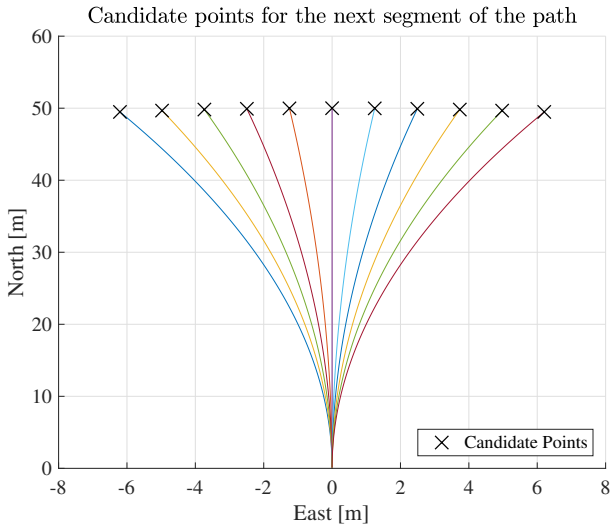


Figure 4.1: An example of how a set of candidate points for the next segment of the path may look like. The number of candidate points is, in this case, eleven, and will depend upon the chosen discretization of the inputs in step 2 of the POP algorithm. The trajectories in the figure are representative for a 3 DOF ASV with constant surge speed and varying yaw speeds chosen from $r = [-0.05 : 0.01 : 0.05]$ [rad/s].

numerically solving the deterministic differential equation. One will then obtain respective candidate points for each possible input. Figure 4.1 exemplifies this notion. In this case, an ASV has been modelled in accordance to section 3.3 and the yaw speeds have been discretized to give a set of eleven possible candidate points.

The remaining steps rely on the creation of the PDFs that will output the probability of reaching a certain position, say (x, y), for each candidate point. Recall that each candidate point corresponds to one of the discretized inputs. This part is vital for the POP algorithm, as it is used to determine the probability of reaching the target position (x_{tar}, y_{tar}) from a candidate point. Unlike the explanation of a probability density function in definition 3.2, this density function will be two dimensional and will therefore be referred to as $p(x, y)$. However, the properties listed in definition 3.2 still apply. In order to determine $p(x_{tar}, y_{tar})$, the PDF must be built using the solutions of the SDEs from step 4.

The Euler-Maruyama method is used to solve the SDE N_{SDE} times, exemplified in figure 4.2 where three trajectories are shown with $N_{SDE} = 10$. In order to generate a trajectory for each candidate point, the initial value in the Euler-Maruyama method is equal to the the candidate point under investigation. So, in equation (3.41), X_i when $i = 0$ is set to the co-ordinates of the current candidate point in NED. From there, the stochastic behaviour of the SDE will ensure that the trajectories will vary for each of the solutions, a characteristic that is visible in figure 4.2. Thus, a set of end-points (x_k, y_k) for each of the N_{SDE} trajectories

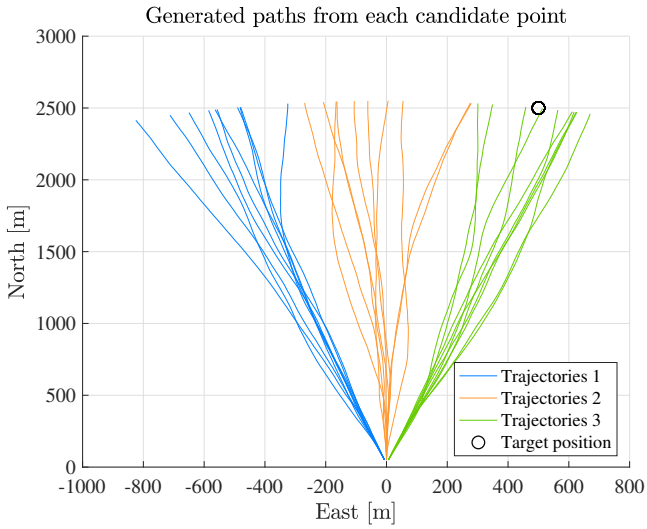


Figure 4.2: Following the example candidate points from figure 4.1, the corresponding solutions to each SDE is visualized in this figure. In this example, $N_{SDE} = 10$ and there are three candidate points corresponding to one yaw speed chosen from $r = \{-0.05,\ 0,\ 0.05\}$ [rad/s].

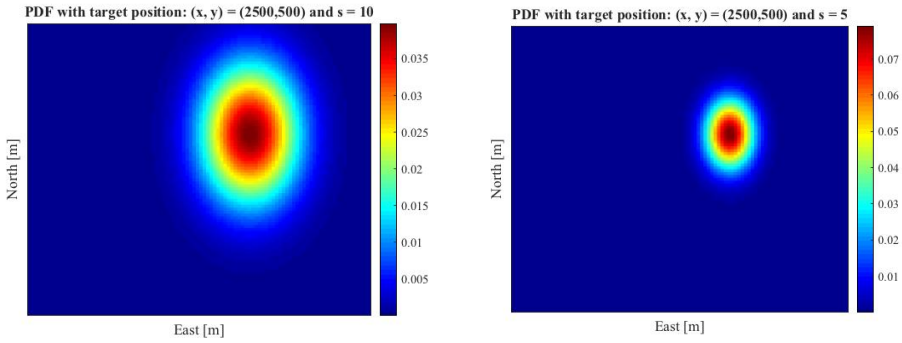
originating from the candidate point at hand will be stored and used in step 5.

The creation of the PDF, in step 5, is based on the assumption that the vessel's position in NED, $(N, E) = (y, x)$, follows a normal distribution. For each candidate point, a PDF is built by summing up the normal distributions for each of the trajectories generated by solving the SDE in step 4, as shown by equation (4.1). The end-points (x_k, y_k) of each trajectory are used as the mean of the normal distribution, evident by comparing the expression in equation (4.1) to (3.20). For each solution of the SDE, the PDF is equal to that of a two-dimensional normal distribution and is given by equation (4.2). In the equation, s refers to the standard deviation.

$$p(x, y) = \frac{1}{N_{SDE}} \sum_{j=1}^{N_{SDE}} g_j(x, y) \tag{4.1}$$

$$g_j(x, y) = \frac{1}{\sqrt{2\pi s^2}} \exp\Big(- \frac{(x - x_k)^2}{2s^2} - \frac{(y - y_k)^2}{2s^2} \Big) \tag{4.2}$$

The resulting PDF $(p(x, y))$ is visualized using a heatmap over the state-space in figure 4.3. The standard deviation (s) should be set to match how the vessel's positions tend to spread out due to the stochastic effects. For ASVs, the stochastic effects include environmental disturbances, sensor noise and model uncertainty; all



(a) **(b)**

Figure 4.3: The two PDFs show how the final density function $p(x, y)$ changes as the standard deviation s is altered, where $s = 10$ in (a) and $s = 5$ in (b)

causing the vessel's response to deviate for the same set of inputs. These effects are difficult to quantify without access to a model ship and experimental values, hence, the standard deviation of the normal distribution in equation (4.2) is estimated. The effect of varying the standard deviation (s) is evident in figure 4.3, where a larger value for s, $s = 10$, results in a greater spread in the PDF and overall lower probability values compared to $s = 5$.

Once the PDFs for each candidate point are created, the target position (x_{tar}, y_{tar}) is input to the PDFs ($p(x_{tar}, y_{tar})$) to give the probability of reaching the target for each candidate point. This information is used in step 6, where the candidate point with the largest probability is chosen as the next waypoint in the path. When this is completed, only step 7 remains; to check whether or not the chosen candidate point is in close proximity to the target point. If so, the POP algorithm is terminated and the list of waypoints generated is used by the GNC system to steer the ASV. If not, the procedure is repeated for a new set of candidate points.

As the global path planning problem involves static obstacles, POP needs to include an approach to handle and steer away these obstacles. This procedure is outlined in the next section.

4.1.1 The POP Algorithm with Static Obstacles

The procedure explained above works well in an environment without obstacles, however, static obstacles must be considered in global path planning. In the POP algorithm, this is completed by terminating the trajectories for each solution of the SDE once they intersect with a static obstacle, as illustrated in figure 4.4. This will then decrease the probability of reaching the target point for the respective candidate point, as the end-points (x_k, y_k) are located further from the target point. As a result, the chance that a candidate point on collision course with a

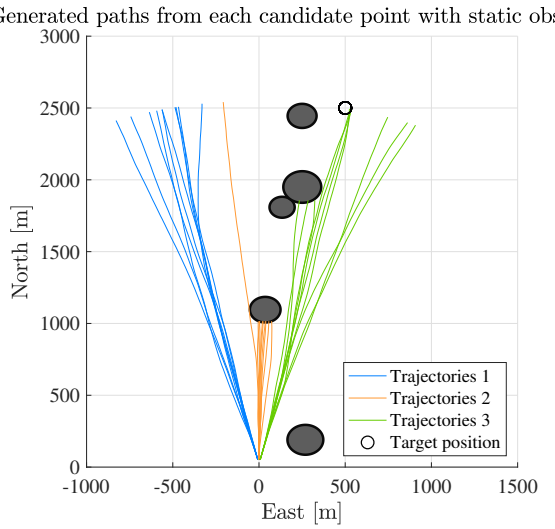


Figure 4.4: Following the parameters in figure 4.2, the solutions of the SDE corresponding to each yaw speed is visualized with the inclusion of static obstacles. It is clear how the trajectories are cutoff once they reach an obstacle.

static obstacle is chosen as the next waypoint is reduced. This procedure was first outlined in [34], where one static obstacle is considered. The performance of this approach in an environment with *multiple* static obstacles is investigated in this thesis, and results are presented in chapter 6.

4.2 Algorithm for Global Path Planning

This section covers how the POP algorithm can be used in combination with an A* search to find a global path for an ASV. The reader is referred to section 3.6 and 3.5 to read up on the theory and implementation details of the A* search and network representation.

The output of the POP algorithm is a set of waypoints, where each waypoint is chosen from a set of candidate points such that the probability of reaching the target is maximized. This result can be combined with an A* search by discretizing the search environment such that it can be represented as a network. The A* algorithm will then search through the nodes in order to find the path with the lowest cost. Hence, by including the probability of reaching the target into the cost calculation of each node, it is possible to merge POP with the A* search.

The total cost of each node $f(n)$ is determined according to equation (3.42), where the cost-to-come $g(n)$ is an accurate description of the cost to reach the node from the start node whereas the cost-to-go $h(n)$ is an *estimate* of the minimum cost to

reach the target node. As the probability of reaching the target is also an estimate, the results from the POP algorithm are included in the heuristic function $h(n)$, also referred to as the cost-to-go $h(n)$. However, the design of the heuristic has to be completed with care. As mentioned in section 3.6.2, an overestimation of the cost-to-go can lead to paths that are sub-optimal. When optimality is key, it is therefore important to design the heuristic such that it is *admissible*; meaning that it underestimates the cost to reach the target. The design of the total cost $f(n)$ such that it includes POP is explained in the section below.

4.2.1 Design of the Total Cost Function

The total cost function $f(n)$ in the A* search depends upon the heuristic $h(n)$ and the cost-to-come function $g(n)$. The $g(n)$-score of a node is set to the euclidean distance from the initial position of the ASV to the position within the current node, shown in equation (3.43). The main contribution to $f(n)$ in this thesis regards the design of $h(n)$, where POP is among the attributes used to calculate the heuristic cost for the nodes within a network.

The effects of a poorly designed heuristics are listed under section 3.6.2, where the hazard of underestimating or overestimating the heuristic function will depend on the problem at hand. If it is more important to have a short and efficient path towards the target point, then underestimating the true cost is favourable. On the other hand, if computational time is a key factor and the need for optimality is relaxed, then overestimating the true cost of the heuristic is not as critical. These two scenarios can be adapted to motion planning, where optimality is favoured in global path planning whereas computational time is a key factor in collision avoidance.

According to [23], the behaviour of the A* search can be controlled by the heuristic function. The design of the heuristics in the motion planning problem can take many forms, in this thesis however, the heuristic function is varied to have contributions from the factors listed below.

Euclidean distance:
The heuristic is based on the Euclidean distance from the node position towards the target position, shown in equation (4.3). All of the heuristic functions tested in this thesis will include the Euclidean distance ($e_h(n)$).

POP algorithm:
The POP algorithm outputs the probability of reaching the target (p) from a given position, this measure is used to increase the Euclidean distance with a penalty factor that is proportional to the probability of *not* reaching the target.

Safety radius:
The vessel is equipped with a safety radius around its body, in which obstacles

are not allowed to enter. If a node violates the safety radius, a penalty factor is added to its heuristic.

In the design of $h(n)$, the Euclidean distance ($e_h(n)$) in equation (4.3) is the basis for the cost-to-go. In order to reduce the optimality of a node, and thereby increasing its cost, penalty factors are included. These penalty factors mirror the elements listed above, i.e POP and the vessel's safety radius. Each penalty factor will be multiplied with $e_h(n)$, such that $h(n)$ and $g(n)$ both have the same units. The heuristic function is shown in equation (4.4), where the effect of the penalty factors $p = [p_{POP},\ p_{RAD}]$ is made clear. All penalty factors are in the range between 0 and 1. Specifically, $p_{POP} = (1 - p(n))$ where $p(n)$ is the probability of reaching the target position from a node n, such that $0 \leq p_{POP} \leq 1$. As for p_{RAD}, its value is either $p_{RAD} = 0$ or $p_{RAD} = 1$, depending on if the safety radius is violated or not. As a result, the maximum addition to the Euclidean length is equal to two times $e_h(n)$.

$$e_h(n) = \sqrt{(x - x_t)^2 + (y - y_t)^2} \tag{4.3}$$

$$h(n) = e_h(n) + (p_{POP} + p_{RAD})e_h(n) \tag{4.4}$$

This addition of multiples of $e_h(n)$ to $h(n)$ means that the heuristic cost influences the value of the total cost $f(n)$ to a much higher extent than the cost-to-come $g(n)$. [23] mentions how it is important that $h(n)$ and $g(n)$ are within the same scale, such that there is an equilibrium in how each cost functions influences the total cost. In order to increase the value of $g(n)$ from equation (3.43) to the same level as $h(n)$, $g(n)$ is re-defined according to:

$$\beta = \{0, 1, 2\} \tag{4.5}$$

$$e_g(n) = \sqrt{(x - x_s)^2 + (y - y_s)^2} \tag{4.6}$$

$$g(n) = e_g(n) + \beta e_g(n) \tag{4.7}$$

where equation (4.5) shows the possible values for β in equation (4.7). The value for β depends on the included penalty factors in $h(n)$. The various design for $f(n)$ that are tested are listed in chapter 6, where the optimality of each design is determined by the means of simulations of a global path planning scenario.

4.3 Algorithm for Collision Avoidance

The problem formulation in a collision avoidance scenario differs from that in global path planning. In global path planning, the target position is well defined and the obstacles are static, as a result it is not necessary to consider the time-variable in

the planning process. It is however possible to do so, but this thesis is operating with an ASV that maintains a constant forward speed. Contrarily, time is an important aspect in collision avoidance. Further, the target position is not a well-defined constant once a collision situation occurs. The ASV has to deviate from its global path, and follow an evasive maneuver generated by the collision avoidance path planner. This section investigates how POP and the A* algorithm are used in unison to generate a COLREGS compliant evasive maneuver.

Currently, the POP algorithm has only been tested with one static obstacle. Thus, the extension to handle *dynamic* obstacles is developed in this thesis and has not been tested before. It is a preliminary method, that focuses on generating a procedure for incorporating POP into the A* search when a collision situation is detected.

The general outline of the proposed collision avoidance technique is as follows:

Step 1: Use CPA to determine risk of collision
Once a target vessel is detected, the closest point of approach (CPA) is used to calculate if there is a risk for a collision. If that is the case, the algorithm proceeds to step 2.

Step 2: Determine the COLREGS situation
The relative bearing (β) is used to classify which COLREGS rules apply to the current collision situation.

Step 3: Generate an evasive maneuver
The A* search with the POP algorithm is used to generate an evasive maneuver. The local path is created such that it lasts until the collision is successfully avoided.

Step 4: Converge back to the global path
Once the target vessel is avoided, the ASV should return to its original, global path.

With steps 1 and 2 previously outlined in section 3.2, steps 3 and 4 are explained in further detail in this section.

In step 3, A* and POP are used in the same manner as in the global path planning problem, outlined in section 4.2. However, the design of the total cost function $f(n)$ differs. Further, even though the POP algorithm remains unchanged, the location of the target point used in each POP iteration is altered. This adjustment of the target points will be referred to as the *virtual* target point (VTP) method. In step 4 of the procedure, an A* search with the POP algorithm is initiated towards the original waypoint. The cost functions of this A* search are the same for step 4 and 3, and are explained in the section below. The VTP method is then outlined in the final section of this chapter. The implementation details of the collision avoidance algorithm are described in chapter 5.

4.3.1 Design of Total Cost Function

Rather than repeating much of the information regarding the design aspect of the total cost function $f(n)$, the reader is referred to the preliminary paragraphs of section 4.2.1; where the considerations in the cost function design are explained.

The expression for $f(n)$ is presented in equation (3.42), and is repeated in equation (4.8) for reference. As for the cost-to-come $g(n)$ and cost-to-go $h(n)$ functions in the A* search for a collision avoidance scenario, the final expressions are shown in equations (4.9) and (4.10).

$$f(n) = g(n) + h(n) \tag{4.8}$$
$$g(n) = 0 \tag{4.9}$$
$$h(n) = (1 - p(n)) = p_{POP}(n) \tag{4.10}$$

The definition of p_{POP} in equation (4.10) is the same as previously explained in global path planning, such that the heuristic function in collision avoidance is equal to the probability of *not* reaching the virtual target point.

Notice that the Euclidean distances are removed from the $g(n)$ and $h(n)$ functions, compared to the global path planning cost functions. This is implemented as the aim of this thesis is to investigate how the *POP* algorithm can generate collision avoidance paths. In the global path planning problem however, the addition of Euclidean distances were necessary to obtain conclusive results.

4.3.2 Virtual Target Points (VTPs)

The POP algorithm relies heavily on a well-defined target point to guide the procedure in the correct direction. The method of virtual target points (VTPs) is therefore created to ensure that the collision avoidance algorithm with POP generates safe paths, where the term "safe" refers to adhering to COLREGS and maintaining a safe distance between the ASV and the target vessel. As mentioned in the general outline of the collision avoidance algorithm on page 57, CPA and relative bearing (β) are used to determine if there is a collision risk and classify which COLREGS rules apply. The reader is referred to section 3.2, where the steering and sailing rules applicable to an ASV are listed. Below is an explanation of how the VTP placement proceeds for each COLREGS scenario where the ASV is a give-way vessel, i.e crossing from the right, head-on, and overtaking. Each of these scenarios are later combined to test the developed collision avoidance algorithm. The results are presented in chapter 6.

Note that the target vessels follow a constant velocity model with a turn rate equal to zero. Further, the positions and velocities of the target vessels are assumed to be perfectly known. Such that no uncertainties are incorporated into their projected positions and velocities.

Rule 16: Crossing from the Right

In order to obey rule 16 in COLREGS, the VTPs have to placed to ensure that the ASV passes behind the target vessel in a crossing from the right situation. Figure 4.5 shows the general procedure for placing the VTPs once the collision risk is detected at time $t = t_0$. In a crossing from the right situation, the VTPs move as time passes. This detail is included as the motion of the target vessel, labelled as "B" in the figures, means that placing the VTP behind vessel B in $t = t_0$ would lead to an unnecessarily large avoidance maneuver.

As each collision situation differs, two variables called α and γ are included to ensure that the placement of the VTPs is optimal. Figures 4.5a and 4.5b show how α and γ affect the VTP placement. The value of α decides the placement of the initial VTP, and which projected position of the target vessel it should follow. Two examples of different values for α are included in figure 4.5a, where $\alpha = 1$ means that the initial VTP is placed behind vessel B at the time instance $t = t_0$. Whereas an $\alpha = 0.5$, means that the initial VTP is placed behind vessel B at time $t_0 \leq t \leq t_{cpa}$, where the relative velocities of A and B decide the exact positioning. If the two vessel have the exact same velocity, then $\alpha = 0.5$ coincides with halfway between $t = 0$ and $t = t_{cpa}$, exemplified in figure 4.5b. The exact location of the initial VTP is given by equation (4.12), where $p_{B,1}$ from equation (4.11) is the projected position of p_B and r_B is the safety radius around vessel B.

$$p_{B,1} = p_{B,t_0} + \alpha(v_B \times t_{cpa}) \tag{4.11}$$

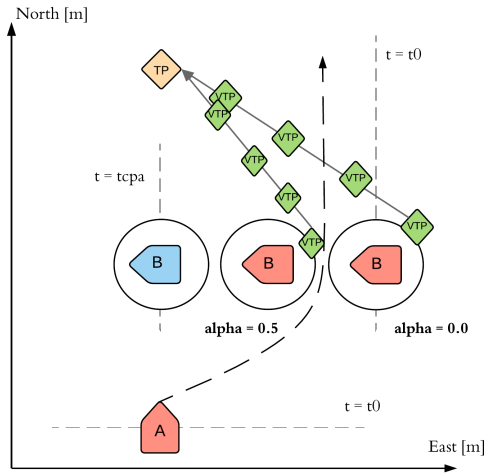
$$VTP_1 = p_{B,1} + r_B \tag{4.12}$$

The variable γ on the other hand, decides how the VTPs are moved back towards the original waypoint of the global path. In figure 4.5b, "TP" refers to the global waypoint. When $\gamma = 1$, the VTPs move back towards TP such that the final VTP is equal to the global TP when $t = t_{cpa}$. If γ is reduced, and $\gamma = 0.5$ as exemplified in figure 4.5b, then the VTPs move towards the TP in half the speed they would have moved with if $\gamma = 1$. The final set of VTPs throughout the collision scenario in a crossing from the right situation are defined recursively, as shown by equation (4.14), where ΔVTP represents the necessary increments to move VTP_1 towards the global waypoint $TP = WP_A$.

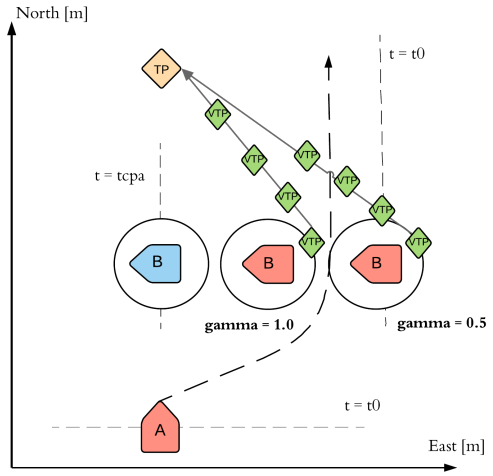
$$\Delta VTP = \frac{WP_A - VTP_1}{N_t - 1} \tag{4.13}$$

$$VTP_{i+1} = VTP_i + \gamma \times \Delta VTP \tag{4.14}$$

Note that the effect of increasing both α and γ means that the ASV, labelled as "A" in figure 4.5, passes the vessel "B" at a shorter distance. Further, both variables are defined in the range between 0 and 1.



(a)



(b)

Figure 4.5: Placement of the VTPs in a crossing from the right situation, where (a) shows how the parameter α affects the VTP placement whereas (b) shows how the parameter γ affects the VTP placement.

Rule 14: Head-on

According to rule 14, *both* vessels are to change their course to starboard. This behaviour is achieved by the following placement of the VTPs.

Unlike the VTPs in a crossing situation, the VTP in a head-on situation is constant. This approach is used as the two vessels are on reciprocal courses, such that moving the VTP to the right of the target vessel, labelled as vessel "B" in figure 4.6, is sufficient for making the ASV changes its course to starboard. As for the exact location of the VTP, it is defined relative to the projected position of vessel B at time $t = t_{cpa}$, shown in equation (4.15). The placement of the VTP is calculated according to equation (4.16), where r_B refers to the safety radius around the target vessel.

Much like the previous VTP placement, the positioning of the VTP can be adjusted with a variable, called κ in a head-on situation. The value of κ decides how far to starboard the ASV should alter its course, such that a large value for κ creates a larger evasive maneuver than smaller values. Further, κ should not be lower than 1, as this places the virtual target point within the safety radius of the target vessel.

$$p_{B,t=t_{cpa}} = p_{B,t=t_0} + v_b \times t_{cpa} \tag{4.15}$$

$$VTP = p_{B,t=t_{cpa}} + \kappa \times r_B \tag{4.16}$$

Rule 13: Overtaking

Unlike the head-on and crossing situations, the overtaking rule in COLREGS allows for course changes to port and starboard. In figure 4.7, the course change of the ASV is set to the starboard side. However, it could just as well pass vessel B on the port side.

By comparing figures 4.7 and 4.6, it appears that the placement of the VTP in an overtake situation is almost identical to that in a head-on situation. This is the case, the only difference is that the sign of κ can be altered, depending on which side of vessel B the ASV wishes to pass. Other than that, the placement of the VTP is constant, and follows the set-up provided by equations (4.15) and (4.16). The variable κ has the same interpretation in overtaking as in head-on collision scenarios.

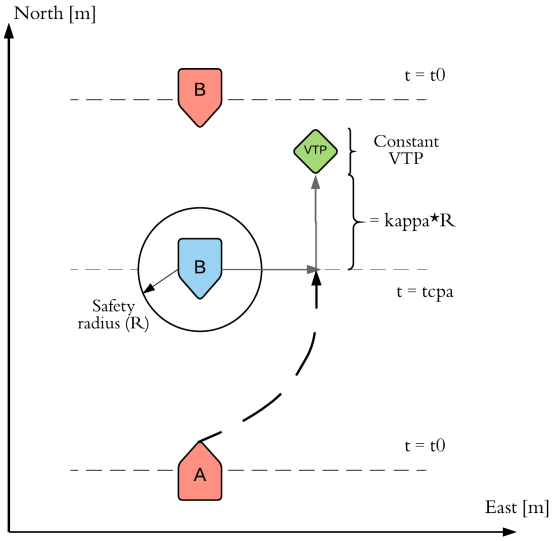


Figure 4.6: Placement of the VTPs in a head-on situation. The only parameter affecting the VTP placement is κ.

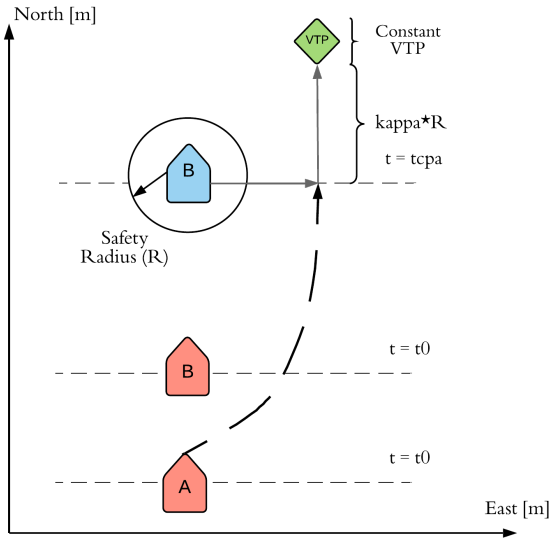


Figure 4.7: Placement of the VTPs in an overtaking situation. The only parameter affecting the VTP placement is κ.

Chapter 5

System Implementation

This chapter will cover how the total system, consisting of a motion planning system and a GNC system, is implemented in MATLAB and Simulink. The simulator is created with the intention to study the limitations and feasible motions of an ASV. These simulations are then used to ensure that the generated path in the global and local path planner is feasible. This point is stressed throughout this thesis, as it is vital to incorporate the vessel's dynamics into the motion planning algorithms.

First, an overview of the general assumptions are listed. These are then followed by an outline of the details concerning the simulator development. The remaining sections cover the implementation of the motion planning algorithm, first detailing global path planning and then moving on to collision avoidance. These implementation details are important to understand the contributions and limitations of this work.

5.1 General Assumptions

In order to comprehend the design of the motion planning algorithm, the basic assumptions used to simplify the problem are stated.

With regard to the ASV model, two main assumptions are made. First, it is assumed that the ASV has a constant forward speed U, where the velocity in sway is negligible such that $U = \sqrt{u^2 + v^2} \approx u$. Next, the possible movements of the ASV are discretized according to three yaw speeds, $r_k = [r_1,\ r_2,\ r_3]^T$. The numerical value for forward speed is summarized in table 5.1, whereas the discretized values for the yaw speeds depend upon the map representation and the feasibility design. The relevant r_k values are therefore given in section 5.3.1.

As for other assumptions, it is assumed that the there are no environmental disturbances. This is completed in order to simplify the motion planning algorithm,

Table 5.1: Constant speed (U) used within this thesis.

Parameter	Value
U $[m/s]$	1

such that the performance of the motion planner is measured in accordance to how it calculates a new path, not as to how it handles environmental disturbances. Handling environmental loads is obviously a criteria for any motion planner, but is left for further work in this thesis.

As for the collision avoidance module, it is assumed that the target vessels are *non*-reactive; meaning that the target vessels follow predicable paths. Further, the information about the target vessels is assumed to be perfectly known. This differs from the static obstacles, where the uncertainty from the environment and sensor information is integrated into the global path planning algorithm. For the collision avoidance algorithm however, focus was kept on creating an algorithm that uses POP such that evasive maneuvers adhering to COLREGS are created. The addition of target vessel uncertainties is left for further work, and discussed in chapter 9.

5.2 Simulator Development

A simulator is created in MATLAB and Simulink to analyze the set of feasible motions the ASV model can follow. This section takes a look at the ASV model, and development of its guidance and control systems.

5.2.1 The Model ASV

In recent years, plans regarding large, battery powered ASVs have been launched. These include DNV GL's ReVolt and Yara Birkeland, which is a collaboration between Yara, Kongsberg, DNV GL, Marin Teknikk, SINTEF Ocean and the Norwegian maritime authorities. ReVolt is intended to have a length of 60 [m] in full scale [52], and Yara Birkleand is set to be 79.5 [m] [4]. However, before ASVs of this scale can be launched at sea, testing is required. For this reason, most current ASVs are of a smaller scale. Take the small scale models for ReVolt and Yara Birkeland, with lengths of $3m$ and $6m$ respectively.

With this in mind, the model ASV used in the simulator is of a smaller scale. It has a length of 2 [m], and the following 3 DOF system matrices:

$$\mathbf{M} = \begin{bmatrix} 76.6539 & 0 & 0 \\ 0 & 149.4620 & -1.0319 \\ 0 & -1.0319 & 34.0793 \end{bmatrix} \tag{5.1}$$

$$\mathbf{D} = \begin{bmatrix} 12.2033 & 0 & 0 \\ 0 & 11.8710 & 0.5852 \\ 0 & 0.5852 & 4.3710 \end{bmatrix} \tag{5.2}$$

The parameters were estimated using MCSim CyberShip 3.

As motion planning is a high-level task, the model used for the ASV is kept as simple as possible. Hence, the rigid-body and added-mass Coriolis-centripetal matrices are neglected, along with the non-linear damping terms. This simplification reduces the 3 DOF model in section 3.3 to the one presented in equations (5.3) and (5.4), where $\boldsymbol{\eta} = [N,\ E,\ \psi]^T$ and $\boldsymbol{\nu} = [u,\ v,\ r]^T$. As the simulator is created with no environmental disturbances, then the dynamical model presented in section 3.3 is simplified by $\boldsymbol{\tau}_{env} = \boldsymbol{\tau}_{wave} + \boldsymbol{\tau}_{wind} = 0$. The actuator forces are represented by $\boldsymbol{\tau}$, and are explained in the next section.

$$\dot{\boldsymbol{\eta}} = R(\psi)\boldsymbol{\nu} \tag{5.3}$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}\boldsymbol{\nu} = \boldsymbol{\tau} \tag{5.4}$$

5.2.2 Actuator Models

The actuators for the ASV model include a rudder and a propeller, where both are located at the stern of the vessel at the center line. The reader should note that the actuator models are developed from a control perspective, thereby lowering the need for detailed modelling. As a result, only the main physical aspects of the actuators are captured by the following models.

From [20, p. 398], the control force from a propeller or a rudder can be written as:

$$F = ku \tag{5.5}$$

where u represents the control input, which is shaft speed n [rad/s] for a propeller and rudder angle δ [rad] for a rudder; and k is the force coefficient for the actuator in question.

For the propeller model, the generated thrust $F = \tau_{prop}$ is affected by many parameters. Using the notation and theory from [49], the thrust can be formulated

as a function depending on shaft speed n, in rotations-per-seconds [rps]; time-varying states ($\boldsymbol{x}_p$), such as pitch ratio, advance velocity and submergence; and fixed thruster parameters ($\boldsymbol{\theta}_p$), such as propeller diameter, geometry and position:

$$\tau_{prop} = f_T(n, \boldsymbol{x}_p, \boldsymbol{\theta}_p) \tag{5.6}$$

Keeping in line with [49], a quadratic relationship can be used to model the relationship between the shaft speed n and the thrust force; resulting in the expression in equation (5.7). K_T is the thrust coefficient, ρ is the density of water and D is the propeller diameter.

$$\tau_{prop} = K_T \rho D^4 sign(n) n^2 \tag{5.7}$$

For the sake of the ASV model in Simulink, all the constants, i.e ρ, K_T and D, are collected in one constant, called K_{prop}. The conversion of n from rps to rad/s, $n\ [rps] = 2\pi n\ [rad/s]$, is also included in K_{prop}. With this in mind, the resulting expression for the thrust is given by:

$$\tau_{prop} = K_{prop} sign(n) n^2 \tag{5.8}$$

For the rudder, foil theory from [50, p. 44] is used to develop the actuator model. The rudder on the ASV model is a conventional rudder, which is a foil connected to the hull of the vessel. Using linear foil theory. the sway force $F = \tau_{rud}$ generated by the rudder can be formulated as a function of U, the forward speed; ρ, the water density; c, the cord length; and δ; the rudder deflection (angle of attack for the foil) in radians:

$$\tau_{rud} = \pi \delta \rho U^2 c \tag{5.9}$$

Much like the actuator model of the propeller, the constants in the expression for τ_{rud} are collected in one constant, called K_{rud}. Unlike τ_{prop} in equation (5.8), the relationship between the control input δ and the control force τ_{rud} is linear, evident in equation (5.10). Note, it is assumed that the sway velocity component is approximately zero, $v \approx 0$, as the ASV will have a heading autopilot to keep the vessel on its course. Therefore, $U = \sqrt{u^2 + v^2} \approx u$ in equation (5.10). The numerical values of the K_{prop} and K_{rud} are summarized in table 5.2, along with the saturation levels of the propeller and rudder.

$$\tau_{rud} = K_{rud} U^2 \delta \tag{5.10}$$

Using the geometry of the ASV model, the locations of the actuators, and the actuator models described above; the resulting force vector in each DOF is shown

Table 5.2: Actuator model parameters and saturation levels.

Parameter	Value
K_{prop}	2
K_{rud}	6
n_{max} $[rps]$	85
δ_{max} $[^\circ]$	15

in equation (5.11). This is also the vector represented by $\boldsymbol{\tau}$ in (5.4), and is implemented within the ASV model in Simulink.

$$\boldsymbol{\tau} = \begin{bmatrix} F_x \\ F_y \\ M_\psi \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_y l_x - F_x l_y \end{bmatrix} \underbrace{=}_{l_y=0} \begin{bmatrix} \tau_{prop} \\ \tau_{rud} \\ \tau_{rud} l_x \end{bmatrix} \tag{5.11}$$

5.2.3 Low-Level Controllers

In order to properly follow the path from the guidance system, a surge and a heading controller is implemented. To view the details of the tuning process and theory behind each of the controllers, the reader is referred to appendix A. The outputs from the heading and surge controllers are given below, in equations (5.12) and (5.13) respectively. The controller parameters are listed in table 5.3.

$$\delta_c = -K_{p,\psi}\tilde{\psi} - K_{d,\psi}\tilde{r} - K_{i,\psi}\int_0^t \tilde{\psi}(\tau)d\tau \tag{5.12}$$

$$n_c = -K_{p,u}\tilde{u} - K_{i,u}\int_0^t \tilde{u}(\tau)d\tau \tag{5.13}$$

Table 5.3: The low-level controller parameters.

Parameter	Value
$K_{p,\psi}$	3.656
$K_{d,\psi}$	8.576
$K_{i,\psi}$	0.284
$K_{p,u}$	1.00
$K_{i,u}$	0.100

5.2.4 Guidance Controller

Within the simulator, a guidance controller in the form of a line-of-sight (LOS) controller is implemented. This controller is based on the LOS guidance laws, whose theory is explained in section 3.1.3.

The specific LOS guidance law implemented is the lookahead-based steering approach. This approach relies on two parameters, the lookahead distance Δ and the radius of the circle of acceptance R_{accept}, shown in equations (3.3) and (3.4) respectively. According to [20], a rule of thumb is to use the ship's length to set these parameter values, where $\Delta \approx 1.5L_{pp} - 2.5L_{pp}$ and $R_{accept} \approx 2L_{pp}$. Following this advice for Δ and R_{accept} results in the guidance parameter presented in table 5.4. For R_{accept} on the other hand, its value is set to a lower value than the suggestion from [20]. If the guidance controller is going to follow a set of close-spaced waypoints, which will typically occur in a collision avoidance maneuver, the radius of the circle of acceptance must be smaller. Its final value is also presented in table 5.4. Further, an example run with a set of waypoints is presented in appendix B to demonstrate the capability of the guidance controller.

Table 5.4: The guidance controller parameters

Δ [m]	R_{accept} [m]
$1.5L_{pp}$ $= 3$	L_{pp} $= 2$

Note that the effect of currents is neglected within the simulator, as environmental factors are not considered within the motion planner. Therefore, the steering law outputs the necessary *heading* ψ_d to follow a set of waypoints rather than the necessary *course* χ_d. As a zero sideslip angle β leads to $\chi_d = \psi_d$. Further, it is assumed that the ASV model is equipped with a sensor measuring its velocity, enabling the steering law to align the vessel's velocity vector and the LOS vector. However, with no currents implemented, the velocity vector of the ASV is perfectly aligned with its body x-axis. So, in principle, the guidance controller aligns the ASV's body x-axis with the LOS vector, but as there are no currents to disrupt the motion of the ASV, it will also align its velocity vector with the LOS vector. As a result, the ASV will travel in the correct direction towards the given path.

5.3 Global Path Planning Algorithm

The global path planning problem is solved using an A* search in combination with the POP algorithm. This section explores the implementation details of the planning algorithm; focusing on network representation, handling static obstacles and how the POP algorithm is designed to fit to an ASV.

5.3.1 Network Representation

Before the A* search can commence, the search environment has to be represented by a network. This section covers how such a network is designed with nodes and arcs, such that the environment and feasible motions of the ASV are accurately portrayed. First, the set-up of the nodes with regard to discretizing the surrounding environment of an ASV is explained, followed by a feasibility design. This design decides how the arcs are set to connect the nodes within the network. The reader is referred to sections 3.5 and 3.6 for theory regarding networks and the A* algorithm.

Defining the Nodes

The nodes of the network make up the search map. For global path planning, this search map covers the area between the initial position of the ASV and its target position. Each node represents a position in NED for a 3 DOF ASV, $p = [N,\ E,\ \psi]^T$, therefore the term node is used to refer to a position in NED throughout this thesis. The nodes define a squared area, in which the center of the node corresponds to a position (p). Note that time is not an aspect that is considered here, as the world model does not consist of any moving obstacles in global path planning.

Once the positions (p) in the search map are defined according to the nodes, the ASV's heading (ψ), must also be incorporated into the network. This is completed by defining the heading in each node according to the heading of the current node, an idea that is visualized in figure 5.1. The current node in the figure has $\psi = 0°$, whereas its successor nodes can have $\psi = \{0,\ -45°,\ 45°\}$. The value of ψ in the successor nodes depends on the placement of the successor node relative to the current node. The vessels colored in red in figure 5.1 all have the same heading as the current node, whereas the blue and green vessels are turned an additional $\pm 45°$ relative to the current node. This pattern is repeated throughout the search map. Next, follows an explanation of how the successor nodes are placed relative to the current node, i.e how the arcs in the network are defined.

Defining the Arcs

In order to ensure that the path generated by the A* search is feasible, the reachability of the ASV has to be defined. Reachability refers to the nodes the vessel can reach from its current node, i.e the successor nodes in figure 5.1. An ASV's reachability will depend on its dynamic behaviour and other parameters, such as forward speed U. With an assumption of constant forward speed, the arcs in the network are defined according to a set of discretized yaw speeds. In order to keep a relatively low computational time, three possible yaw speeds are used: $r_k = [r_1,\ r_2,\ r_3]$. The values for r_k are assigned such that the the motions are feasible for the ASV model defined in section 5.2.1. Further, the arcs in the network are design according to a *T-neighbourhood*, an idea used to incorporate vessel constraints into an A* search from [5].

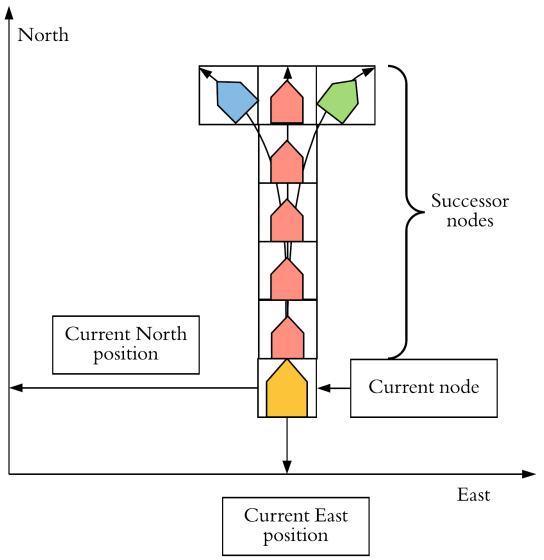


Figure 5.1: The successor nodes from the current node span into the NED-frame in the shape of a T, thus, giving rise to the name *T-neighbourhood* [5]. The heading ψ of the vessel within each successor node depends where in the T-neighbourhood the successor is located *and* on the heading in the current node. The green vessel represents a turn to right, and an additional 45° on the heading; whereas the blue vessel represents a turn to the left, and a reduction of $-45°$ in the heading-

The T-neighbourhood is defined according to figure 5.2a, where the successor nodes are shaded in green. [5] defines the T-neighbourhood according to three parameters, the head, body and node distance, all visualized in figure 5.2a. The size of the head length is normalized to three nodes, and its length in meters will therefore depend upon the chosen node distance (n_d) for the network. The body on the other hand, is adjusted to match the feasible motions of the particular ASV model used in the research. Once the A* search commences, the reachable nodes are continuously defined by the T-neighbourhood. This aspect is presented by figure 5.2b, where two steps into the A* search are shown. From the initial node, the first T-neighbourhood is shaded in green, whereas the second level of the search and its reachable nodes are shaded in yellow.

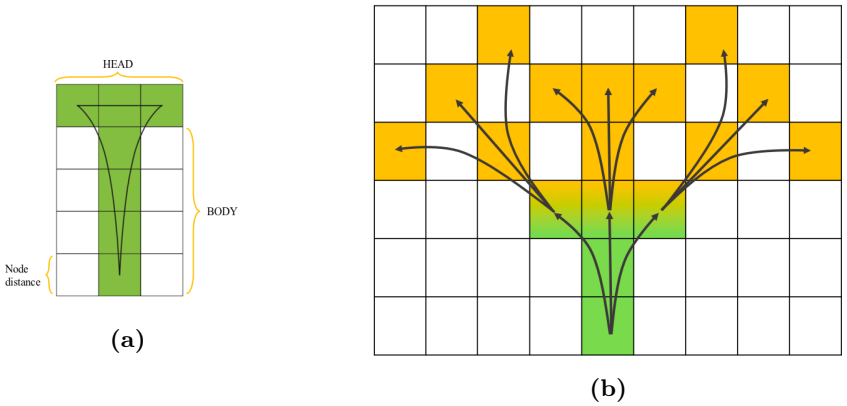


Figure 5.2: (a) Visual representation of the T-neighbourhood. Original figure is obtained from [5]. (b) In the network in the A* search, the successor nodes from each node are defined according to the T-neighbourhood. Here, two steps ahead are visible from the current position. For this example the size of the body is equal to two nodes, whereas the size of the head is three nodes.

The numerical values for the discretized yaw speeds (r_k) are closely related to the design of the T-neighbourhood, as each node in the head represent one yaw speed. Therefore, the size of the body in the T-neighbourhood and the values of the yaw speeds have to carefully chosen. In order to secure that the successor nodes are actually reachable by the ASV defined in section 5.2.1, a simulation using the developed simulator is run with four test cases. Each test case represents one design of the T-neighbourhood, and are summarized in table 5.5. The test case that fits to the ASV's behaviour best, is used as the design of the T-neighbourhood and its yaw speed value is used to set the values for r_1 and r_3 in r_k. Note that $r_2 = 0$ [rad/s], as the successor nodes straight ahead from the current node in the T-neighbourhood have zero turning rate.

From an initial position in the origin and a maximum rudder deflection $\delta_{max} = 15°$, the resulting response of the ASV is shown by figure 5.3. It is clear that tests 1 and 4 are most suitable for the ASV model. In order to keep the size of the body

as small as possible, such that the reachability of the ASV is maximized, test case 1 is chosen as the basis for the T-neighbourhood. Using a node distance of 1 [m], results in the design summarized in table 5.6. Note that the discretized yaw speeds r_1 and r_3 represent the ASV's yaw speed once it reaches the position specified by test case 1.

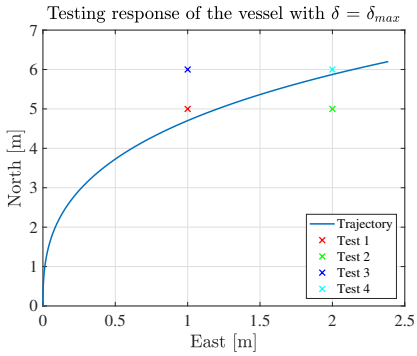


Figure 5.3: The resulting behaviour of the ASV with $\delta = \delta_{max}$. Test cases 1 and 4 are most suitable for the ASV model.

Test #	Δx	Δy
1	1	5
2	2	5
3	1	6
4	2	6

Table 5.5: The four tests used to determine the value for the discretized yaw speeds.

Table 5.6: The design of the T-neighbourhood and the discretized yaw speeds $r_k = [r_1,\ r_2,\ r_3]$

Parameter	Value
Body nodes	5
Head nodes	3
r_1 [rad/s]	-0.05
r_3 [rad/s]	0.05

The chosen design of the T-neighbourhood is verified by a test-run with the waypoints in table 5.7, where each waypoint is placed to match the set-up of the T-neighbourhood. The low-level and guidance controllers in the simulator are used to steer the vessel on the right course. Based on the resulting trajectory in figure 5.4, the ASV clearly manages to follow the waypoints in a satisfying manner. It can therefore be concluded that the arcs in the network, which are defined by the T-neighbourhood, secure feasible paths for the ASV model from section 5.2.1.

Table 5.7: The waypoints used to test the design of the T-neighbourhood.

North [m]	East [m]
0	0
5	1
10	2
15	1

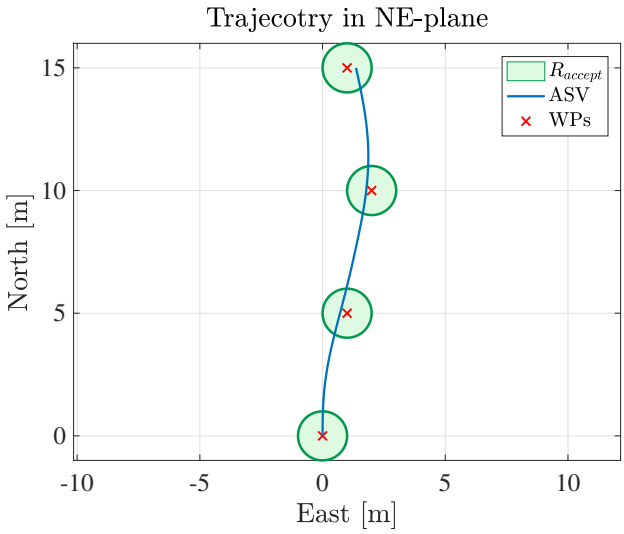


Figure 5.4: The waypoints from table 5.7 are used to test the design of the T-neighbourhood. The resulting trajectory of the ASV supports the chosen design of the T-neighbourhood. Note that R_{accept} refers to the radius of the circle of acceptance from the guidance controller, for reference view section 3.1.2.

5.3.2 Static Obstacles

In the global path planning problem, the world model merely consists of *static* obstacles. For a real-life ASV however, the world model should also include information regarding water depth, currents and so on; as these are environmental factors the motion planner should consider to create an optimal path. Although, that is not within the scope of this thesis.

The generated static obstacles in the simulated environment include uncertainties to reflect the stochastic nature of the ASV's surrounding environment. Take for instance floating obstacles at sea, their position estimates are bound to contain uncertainties due to the effects of varying sea states. Each obstacle is therefore generated by sampling integers from a uniform discrete distribution and is modelled as a circle. Due to the added uncertainties, the location of each static obstacle follows a normal distribution with a mean (μ) equal to the circle's center in the NED-frame, and a standard deviation (σ) proportional to the obstacle's radius. This entails that each obstacle follows a two-dimensional normal distribution $\sim N(\mu = N, E|\sigma = \alpha R)$, where α determines the relationship between the radius and the obstacle's PDF standard deviation. α is tuned accordingly, where figure 5.5 shows how the distribution is altered for $\alpha = 1$ and $\alpha = 2$. For the sake of this example, a one-dimensional distribution is showed, but the effect of increasing or decreasing α will have the same effect on a two-dimensional distribution. Once α is set, the distribution of each obstacle is defined according to figure 5.6, where the one-dimensional distribution is again used for simplicity. The probability ($p(x)$) in the PDF represents an *occupancy probability* for a position x in the NED-frame, in that it measures the probability of being in proximity to a static obstacle. This procedure is based on a similar approach from [35].

The PDF is useful in the POP algorithm as it can represent a safety margin applied to avoid static obstacles. A threshold probability value (p_t) is used to define the margin. For instance, if a path moves too close to the static obstacle and its occupancy probability exceeds p_t then the paths are discarded. This is merged into step 4 of the POP algorithm[1], by terminating the SDE solutions once it hits an obstacle, resulting in a lower probability of reaching the target position for the respective candidate point. With the probabilistic approach for representing obstacles, "hitting" an obstacle means obtaining a probability larger than the set probability threshold in the obstacle's PDF. By using the probability threshold (p_t), it is possible to increase the cost of certain nodes in the A* search and thereby steer away from paths that pass too close to the obstacle.

[1] See page 50

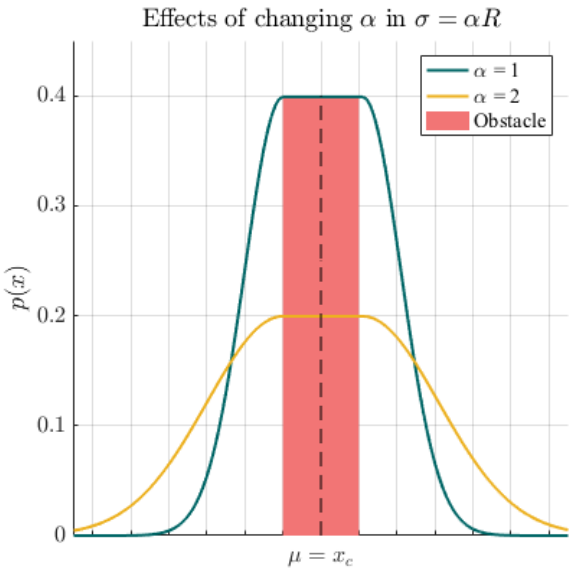


Figure 5.5: The probability distribution for the static obstacles change as α changes. Here exemplified with two values for α in $\sigma = \alpha R$, namely $\alpha = 1$ and $\alpha = 2$.

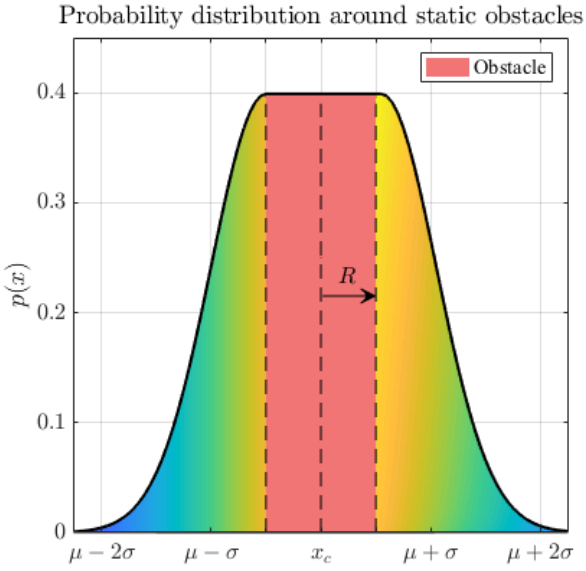


Figure 5.6: An example of how a probability density function (PDF) around static obstacles may look like. The mean of the of the distribution is constant across the space the static obstacle is set to occupy.

5.3.3 Implementation of POP Algorithm

Chapter 4 describes the outline of the POP algorithm for a general path planning problem. This section on the other hand, will clarify how POP has been designed to solve the path planning problem for an ASV. The reader is referred to section 3.4.5 to brush up on the stochastic theory used to develop POP. Following the list of steps on page 50, the implementation details of the POP algorithm are explained one-by-one.

The first step in the POP algorithm is to determine a suitable SDE for the problem at hand. Keeping in line with the idea that motion planning is a high level task, the SDE for the ASV is based on the kinematic relationship in equation (5.3) rather than the kinetics in equation (5.4). The resulting kinematic model of the ASV is therefore given by equation (5.14), which is the kinematic model from equation (5.3) driven by a process noise term ($\mathbf{w}$) that represents the stochastic effects acting on the system. The modelling of $\mathbf{w}$ is completed according to [20, p.294] and is a zero-mean Gaussian white noise processes, and it represents imperfect modelling, disturbances and uncertainties in the system and environment. The addition of the stochastic process ($\mathbf{w}$) results in a solution, $\boldsymbol{\eta}$ in equation (5.15), that is also a stochastic process. As explained in section 3.4.5, a SDE can be represented by an integral and differential form. The resulting SDE based on the stochastic model in equation (5.14) is presented in equations (5.15) and (5.16), representing the integral and differential forms respectively.

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} + \mathbf{E}\mathbf{w} \tag{5.14}$$

$$\boldsymbol{\eta}(t) = \boldsymbol{\eta}_0 + \int_0^t \mathbf{R}(\psi(s))\boldsymbol{\nu}(s)ds + \int_0^t \mathbf{E}\mathbf{w}(s)ds, \quad 0 \leq t \leq T \tag{5.15}$$

$$d\boldsymbol{\eta}(t) = \mathbf{R}(\psi(t))\boldsymbol{\nu}(t)dt + \mathbf{E}d\mathbf{W}(t), \quad 0 \leq t \leq T \tag{5.16}$$

Note that the integral form in equation (5.15) contains an integral with $\mathbf{w}$, this term is replaced by an increment of the Wiener process ($d\mathbf{W}(t)$) in the differential form of the SDE in equation (5.16). This conversion is completed according to the definition of the Wiener process as an integrator driven by white noise[2]. Therefore, the Wiener process related to the integral of the process noise ($\mathbf{w}$) in equation (5.14) is given by:

$$\mathbf{W}(t) = \int_0^t \mathbf{w}(s)ds \tag{5.17}$$

$$\Rightarrow d\mathbf{W}(t) = \mathbf{w}(t)dt \tag{5.18}$$

where equation (5.18) is the increment of the Wiener process used in the differential form of the SDE in equation (5.16).

[2]Refer to section 3.4.4

Being on the topic of SDEs, the implementation details of step four in the POP algorithm is presented before steps two and three. Step four is to solve the SDE in equations (5.15)-(5.16), and is completed using the Euler-Maruyama method described in section 3.4.5. The SDE in the differential form from equation (5.16) is expanded and input into the Euler-Maruyama method from equation (3.41), resulting in the expression in equation (5.19). The labels are used to indicate how the SDE relates to the Euler-Maruyama method in equation (3.41).

$$\underbrace{\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \psi_{i+1} \end{bmatrix}}_{X_{i+1}} = \underbrace{\begin{bmatrix} x_i \\ y_i \\ \psi_i \end{bmatrix}}_{X_i} + \underbrace{\begin{bmatrix} cos(\psi_i) & -sin(\psi_i) & 0 \\ sin(\psi_i) & cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ r_i \end{bmatrix}}_{f(X_i)} \Delta t + \underbrace{\begin{bmatrix} cos(\psi_i) & -sin(\psi_i) & 0 \\ sin(\psi_i) & cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_u & 0 & 0 \\ 0 & \sigma_v & 0 \\ 0 & 0 & \sigma_r \end{bmatrix}}_{g(X_i)} \underbrace{\begin{bmatrix} dW_u \\ dW_v \\ dW_r \end{bmatrix}}_{W(t_{i+1})-W(t_i)} \tag{5.19}$$

The first part of the equation, $X_{i+1} = X_i + f(X_i)\Delta t$, is merely the Euler-method which is used to solve ODEs on the form $\dot{X} = f(X(t))$. In this case, $f(X(t))$ corresponds to the right-hand side of the kinematic model in equation (5.3). The second part however, $g(X_i)(W(t_{i+1}) - W(t_i))$, represents the stochastic effects that disturb the ASV and are modelled using Wiener process increments $dW_i = (W(t_{i+1}) - W(t_i))$.

Definition 3.8 defines the Wiener process, and explains how increments of the Wiener process follow a normal distribution according to $W(t) - W(s) \sim \sqrt{t-s}N(0,1)$. The Wiener process term in equation (5.19) can therefore be sampled from a normal distribution in MATLAB using the function *randn*, explained below. The increment $\delta t = t - s$ in the Wiener process relates to the step-size in the Euler-Maruyama method Δt by an integer multiple $R \geq 1$, such that $\Delta t = R\delta t$. According to [24], this ensures that the simulated Wiener process contains points at which the Euler-Maruyama solution in equation (5.19) is computed. The values for the step-sizes of the Wiener process and the Euler-Maruyama method are summarized in table 5.8.

MATLAB function: randn

```
X = randn
```

The randn function in MATLAB is used to generate a random scalar (X) drawn from a standard normal distribution $\sim N(0,1)$.

As for the $g(X_i)$-term in equation (5.19), the rotation matrix $\mathbf{R}(\psi)$ from the BODY-

Parameter	Value
Δt	0.1
δt	0.01
$\Rightarrow R$	10

Table 5.8: Step-size of Euler-Maruyama method and Wiener process increments.

frame to the NED-frame is included as the process noise ($\mathbf{w}$) acts on the velocities of the ASV. This is evident in equation (5.14), where $\mathbf{w}$ influences the position ($\boldsymbol{\eta}$) of the ASV by affecting its velocities ($\boldsymbol{\nu}$). As a result, the Wiener process terms $[dW_u \ dW_v \ dW_r]^T$ and its strength matrix $\boldsymbol{\sigma} = diag[\sigma_u \ \sigma_v \ \sigma_r]$ in equation (5.19) affect the velocities.

The strength of the Wiener process is portrayed by the matrix $\boldsymbol{\sigma}$ in the $g(X_i)$-term in equation (5.19). In chapter 3, the calculation of the Wiener process' variance is presented and equation (3.30) shows that $Var[W(t)] = \sigma^2 t$. Here, σ stems from the integrated white noise and is its standard deviation, as shown by equation (3.22). This entails that a larger spread in the process noise results in a larger perturbation on the ASV by the stochastic effects, and therefore a larger strength in the Wiener process. Note that it is assumed that there is no correlation between the noise affecting each of the velocities, as the $\boldsymbol{\sigma}$ matrix is a diagonal matrix. To increase the accuracy of the stochastic model, it is possible to neglect this assumption and have a non-diagonal $\boldsymbol{\sigma}$ matrix. However, in order to maintain a degree of simplicity, the diagonal $\boldsymbol{\sigma}$ matrix in equation (5.19) is chosen to represent the Wiener process strength. The values within the matrix are estimated according to the system at hand, and are explained in further detail in section 5.3.4 regarding parameter estimation within the POP algorithm. Once the values for $\boldsymbol{\sigma}$ are set, the SDE used to portray the state transitions for the ASV model and their solutions is complete and the remaining steps of the POP algorithm are initiated.

The second and third steps of the POP algorithm regard discretizing inputs to the SDE and determining the respective candidate points for the next segment of the path. However, as the POP algorithm in the motion planner is combined with an A* search, these steps are already completed. The A* search requires the surrounding environment of the ASV to be represented as a network. Therefore creating the network with a set of nodes, representing the search map in MATLAB and explained in section 5.3.1; and arcs, where each arc is one of three discretized yaw speeds and explained in section 5.3.1, entails that the discretization of the inputs and determining candidate points is already completed in the design of the network of the A* search.

Moving on to step 5, which is the creation of the probability density function (PDF) used to calculate the probability that a candidate point will reach the target position. Based on the solutions of the SDE from step 4, where a total of $N_{SDE} = 10$ solutions for each candidate point are included, the resulting PDF is then created

using equations (4.1) and (4.2). As evident by equation (4.2), the estimate of the PDF requires the variance s^2 to be set. The estimate of the variance is explained in detail in section 5.3.4. By inputting the co-ordinates of the target position (x_t, y_t) into the PDF $(p(x, y))$ from equation (4.1), the probability of reaching the target position from each candidate point is determined. Note that each candidate point has its respective PDF.

Step six concerns choosing the optimal candidate point with regard to the estimated probability. However, as the POP algorithm in the global path planner is used within an A* search, the optimal candidate point is not necessarily based on the largest probability of reaching the target. Other factors such as path length and the vessel's safety radius will also affect the optimality of the candidate point. Section 4.2.1 goes through how the heuristic function in the A* search is defined such that the POP algorithm has an effect on the optimality of each node (also called candidate point). Finally, the termination criteria will also follow the set-up of the A* search presented in algorithm 1. Now, moving on to the parameter estimation that is necessary to design the POP algorithm for ASVs.

5.3.4 Parameter Estimation for the POP Algorithm

Numerous parameters must be set in the POP algorithm to ensure accurate and realistic results. The parameters that are estimated in the implementation are the following:

Standard deviation (s) in p(x,y):
The probability density function (PDF), $p(x, y)$, gives the probability of reaching a position (x, y) in the NED-frame for a candidate point of the final path. Its standard deviation is tuned according to stochastic behaviour of the system at hand.

Strength of the Wiener process ($\boldsymbol{\sigma}$):
The Wiener process represents stochastic effects that disturb the ASV's motion. Its strength is given by the matrix $\boldsymbol{\sigma}$, and is tuned according to the stochastic effects pertruding the system at hand.

Table 5.9 presents the final values for the estimated parameters of the POP algorithm for an ASV. The remainder of this section explains the reasoning behind the listed values.

Table 5.9: The estimated parameters used in the POP algorithm.

Parameter	Value
s	2
σ_u	0.01
σ_v	0.01
σ_r	0.01

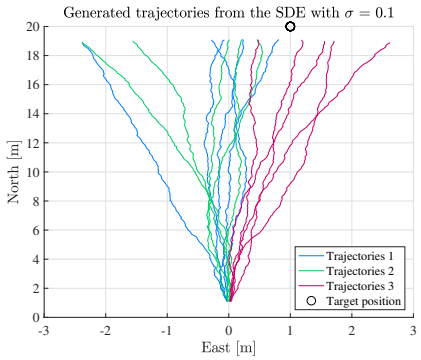
Setting accurate values for s and $\boldsymbol{\sigma}$ should be done experimentally, as completed by [45, 34] in the modelling of a flexible needle and a rolling robot respectively. In [45], numerous experiments are run to collect data of the stochastic effects and the covariance (Σ) of the needle behaviour. Along with an analytical equation for a covariance matrix[3], [45] uses samples from various runs with the flexible needle to find an accurate model and covariance matrix. [34] on the other hand, aims to find an optimal path for a rolling spherical robot using the POP algorithm. In order to verify and determine the strength of the stochastic behaviour of the rolling robot, [34] runs tests to record and study the rolling motion of the robot. In order to accurately determine the strength of the stochastic effects on the ASV, a real-life model is needed to run various tests and then observe how the trajectories vary for the same set of yaw speed inputs. The same goes for the standard deviation (s) of the PDF in equation (4.2) that represents the spread in the final positions of each of the trajectories obtained by solving the SDE $N_{SDE} = 10$ times. However, as this thesis is completed without experimental results, the model parameters s and $\boldsymbol{\sigma}$ must be estimated.

The PDF ($p(x, y)$) gives the probability of reaching a position (x, y) in the NED-frame from a candidate point. Its standard deviation (s) alters the spread of this distribution, and therefore affects the probabilities. As shown by figure 4.3, increasing s means that the overall values for the probabilities at each position (x, y) decreases but the reach of the PDF is increased, in that positions located further from the mean will be given a non-zero probability value due to the increased spread. This is also evident in the equations for $p(x, y)$, (4.1) and (4.2). In [35], the value of s is estimated such that the resulting probability distribution is smooth. A similar approach is used here, resulting in $s = 2$; which is the same value used in [34] and provides reasonable results in the global path planner.

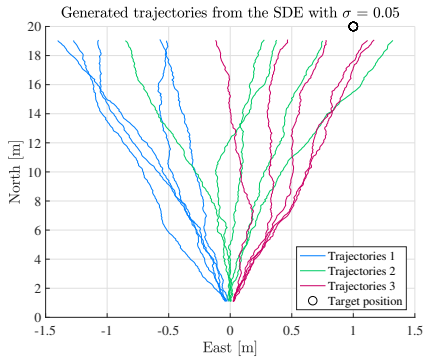
The Wiener process is explained in detail in 3.4.4, and is used in the Euler-Maruyama method from section 3.4.5. The strength of the process is determined by the values in the $\boldsymbol{\sigma}$ matrix, shown in the numerical solution of the SDE in equation (5.19). As no experimental values are available, the values have to be estimated such that they accurately portray the level of stochastic effects on each velocity ($\boldsymbol{\nu} = [u,\ v,\ r]^T$).

As previously stated, the stochastic effects affecting the positions of the ASV in equation (5.14) represent model uncertainties and unknown disturbances. According to [20], the process noise covariance matrix is complex to estimate as it depends on the current sea state, the ASV's heading relative to the environmental forces and the level of uncertainty within the model. Note that the process noise covariance is equal to the strength of the Wiener process $\boldsymbol{\sigma}$. As the simulation environment is created without modelling environmental disturbances, such as waves and currents, it is assumed that the strength of the Wiener process is the same in all DOFs. This entails that $\sigma = \sigma_u = \sigma_v = \sigma_r$. The numerical value for σ is now set such that the response of the ASV is feasible. Figure 5.7 shows how the response of the ASV is altered as σ is altered. From figures 5.7a and 5.7b, it is clear that the strength

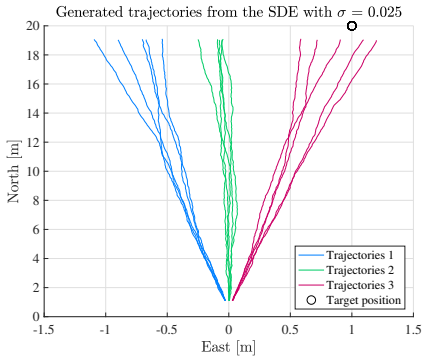
[3]Which is the same as the variance s^2 for $p(x, y)$



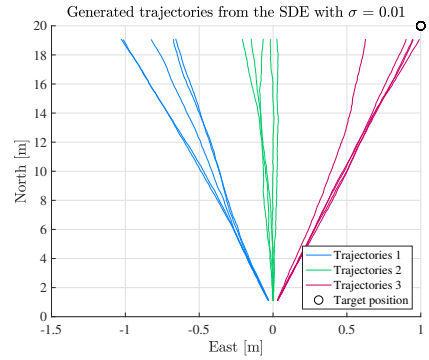
(a) Generated trajectories with $\sigma = 0.1$.



(b) Generated trajectories with $\sigma = 0.05$.



(c) Generated trajectories with $\sigma = 0.025$.



(d) Generated trajectories with $\sigma = 0.01$.

Figure 5.7: The graphs show how the behaviour of the ASV is affected by a set of values for σ. Here simulated with three possible yaw speeds and $N_{SDE} = 5$.

of the Wiener process is too large as the trajectories do not accurately portray an ASV's dynamic behaviour. The trajectories are too random and therefore unrealistic. Figures 5.7c and 5.7d on the other hand, show more realistic trajectories for an ASV. As $\sigma = 0.025$ shows a larger deviation from an expected behaviour of an ASV, the chosen value for σ is 0.01.

5.4 Collision Avoidance Algorithm

The general set-up of the collision avoidance algorithm is quite similar to the global path planning algorithm described above. The motion planning is completed using an A* search, such that the network representation, which includes the feasibility design in the form of the T-neighbourhood; and the implementation of the POP algorithm, which includes the parameter estimation, is unaltered. The methodology

of the collision avoidance algorithm is outlined in section chapter 4, whereas this section focuses on the implementation details of the algorithm.

To avoid repetition, only the key adjustments needed to transform the global path planner to the collision avoidance module are explained in this section. These adjustments are as follows:

Time:
The presence of dynamic obstacles requires the algorithm to keep track of the time. In the A* search, time is included as an additional dimension in the search map. The nodes will then contain information regarding position in NED (N, E), heading (ψ) and time (t).

Dynamic obstacles:
The target vessels are represented as dynamic obstacles, and are modelled according to a constant velocity model with zero turning rate.

Each of these alterations are outlined in further detail below.

5.4.1 Including Time in the A* Search

The search network remains unaltered with regard to the reachability of the ASV, i.e the successor nodes are the same, but time is also included. Much like the heading (ψ), the time (t) of a successor node is defined according to the time of the current node. This recursive definition of the time is depicted in figure 5.8, where turning motion requires more time than motion straight ahead. The example is quite simple, where the node distance is $n_d = 1$ [m], ASV speed is $U = 1$ [m/s] and the initial time is $t_0 = 0$ [s]. With these variables defined, the time increment between each node is calculated according to equation (5.20).

$$\Delta t = \frac{n_d}{U} \tag{5.20}$$

Once each node has its respective time value $(t(n))$, then the aspect of time must be incorporated into the A* search. Further, the search termination criteria from the global path planner cannot be used, as there is no well-defined target node in a collision avoidance scenario. This is solved by using time as a termination criteria, where the time until the closest point of approach (CPA) (t_{cpa}) is used to terminate the search for a minimum cost evasive maneuver. This criteria ensures that the resulting path continues until the collision is avoided. With the target vessels following a constant velocity model and an assumption that they are non-reactive, the target vessel's position in $t = t_{cpa}$ remains equal to its projected position for the CPA. Therefore, by ensuring that the ASV deviates from its respective projected position for the CPA, it is assured that the generated path avoids a collision.

The time of each node is additionally a key parameter to determine its cost, by means of the POP algorithm. Refer to section 4.3.1 to review the cost function

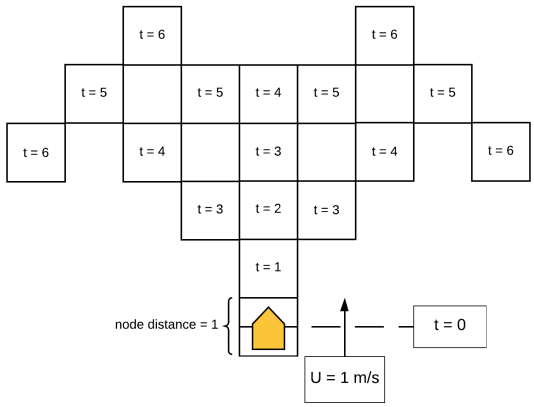


Figure 5.8: Time development in the search map, here exemplified for an ASV at $t = 0$ with a speed $U = 1$ [m/s] and the node distance $n_d = 1$ [m].

of the A* search for collision avoidance. As previously mentioned, the method of *virtual* target points (VTPs) is used to guide the POP algorithm in the desired direction, ensuring that the final evasive maneuver adheres to COLREGS. Each of the generated VTPs are associated to a time instance, such that the probability calculations in POP for a node with time $t(n)$ are completed using the VTP corresponding to the time of the node. This approach ensures that each node is given an accurate cost estimate with regard to the desired target position at each time instance. With accurate cost estimates, a collision avoidance path that follows the VTPs is generated.

In contrast to the global path planner, where static obstacles were represented with an uncertainty, the modelling of the dynamic obstacles in the collision scenario is simplified. This is the subject of the following section.

5.4.2 Dynamic Obstacles

The dynamic obstacles within the collision avoidance module are not directly represented within the search-space for the A* search. Contrarily, the static obstacles in the global path planner were put in the CLOSED list[4] in the initial stages of the search. The dynamic obstacles on the other hand, are indirectly avoided by the incorporation of the VTPs in the cost calculations. The limitations caused by this configuration are further discussed in chapter 7.

As for the motion of each target vessel, it follows a constant velocity model. Equation (5.21) gives its projected position ($p_{TV|t}$) at time t based on a measurement at time t_0 that estimated the vessel's position at $p_{TV|t_0}$. Further, each target vessel

[4]Refer to section 3.6 for definition of CLOSED list

is assumed to have a zero turning rate, such that each vessel follows a straight-line motion. This is an oversimplification of real events, yet a necessary adaption to develop the collision avoidance method with POP, outlined in section 4.3.

$$p_{TV|t} = p_{TV|t_0} + v_{TV} \times (t - t_0) \tag{5.21}$$

Chapter 6

Simulation Results

This chapter presents the results from an array of motion planning scenarios. The first contribution regards global path planning, where the scenario includes only static obstacles; and the second contribution is collision avoidance, where the ASV is set to avoid other vessels in motion. The implementation details of the global path planner and the collision avoidance algorithm is explained in detail in sections 5.3 and 5.4 respectively.

The information for the simulation platform is presented in table 6.1, this platform is used for global path planning and collision avoidance.

Table 6.1: Information regarding the simulation platform.

Computer	MacBook Air (2015)
Operating system	macOS High Sierra
Processor	1,6 GHz Intel Core i5
Memory	8 GB 1600 MHz DDR3
MATLAB version	R2017b

6.1 Global Path Planning

6.1.1 The Scenario

The scenario used to test the path planning algorithm is summarized in table 6.2. In addition to the parameters clarified in section 5.3, two extra parameters were necessary to modify the global path planning algorithm. The first is the safety radius (r_s) around the vessel, this radius defines the area in which obstacles are not allowed to enter and is related to the safety radius penalty factor (p_{RAD}) explained in section 4.2.1. Second, a separate detection radius for the POP algorithm (r_{POP}) was also needed. The r_{POP} radius defines a circle around the vessel where the static obstacles within are accounted for by the POP algorithm. The detection circle is a modification to the POP algorithm, as it was found that the method for handling static obstacles had a tendency of considering too many obstacles. For instance, the POP algorithm would lower the probability of reaching the target from a node if it had an obstacle between its position and the target position, even though this obstacle was located far away. Therefore, the detection radius was introduced, securing that the POP algorithm only steers away from obstacles that are in proximity and make up a danger for the ASV and its current position. The values for r_s and r_{POP} are 2 [m] and 12 [m] respectively.

Table 6.2: Global Path Planning Scenario

Initial Position		Target Position	
North [m]	East [m]	North [m]	East [m]
0	0	30	-43

The target position defined in table 6.2 is located ≈ 52 $[m]$ from the initial position. This distance does not do the *global* path planning problem justice, as this offline path planning is typically completed over large distances. However, the computational time on the simulation platform becomes too large to be able to modify and test the procedure multiple times if the search is completed over longer distances. Further, the algorithm has been tested with a target position located further away, ≈ 500 $[m]$, and the overall behaviour of the planner was similar to its behaviour over shorter distances. Therefore, the relatively short path is used as the scenario to test the ability of the planner in a global perspective.

Implementing the scenario in table 6.2 results in the scenario shown by figure 6.1. The next section shows how the global path planning algorithm outputs an array of paths as the designs of the cost functions in the A* search change.

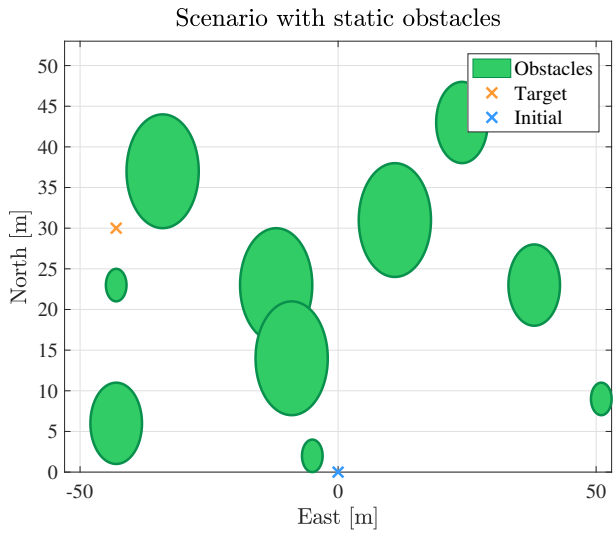


Figure 6.1: The figure shows the map in NED that defines the scenario used to test the path planning algorithm.

6.1.2 The Generated Paths

Various designs for the total cost function are generated and are listed in table 6.3, each of which is tested in an A* search on the scenario in figure 6.1. The total cost function $f(n)$ for the A* search is re-written in equation (6.1) for reference. The aim of these tests is to determine the ideal combination of $h(n)$ and $g(n)$, and investigate how the POP algorithm can be combined with the A* search to solve the global path planning problem.

$$f(n) = h(n) + g(n) \tag{6.1}$$

Note that cases seven and eight in table 6.3 are used as reference values. Case 8 is Dijkstra's Algorithm and is a special case of the A* search, the algorithm is guaranteed to find the shortest path but at a much larger computational time. As for case 7, using $h(n) = e_h(n)$ and $g(n) = e_g(n)$ are the typical function definitions, and are therefore included to see if the penalty factors can improve these results. Further, the cases with the same $h(n)$ functions but different $g(n)$ functions are generated to see the effect of scaling $g(n)$. Thus, ensuring that the contributions from the two cost functions are within a similar scale; where the maximum value for $h(n)$ is set to occur when both $p_{POP} = p_{RAD} = 1$, effectively tripling the Euclidean distance (e_h).

Table 6.3: The various designs of the cost functions, $h(n)$ and $g(n)$, used in the A* search for global path planning.

Case	Heuristic $h(n)$	Cost-to-come $g(n)$
1	$e_h(n) + (p_{POP} + p_{RAD})e_h(n)$	$e_g(n) + 2e_g(n)$
2	$e_h(n) + (p_{POP} + p_{RAD})e_h(n)$	$e_g(n) + e_g(n)$
3	$e_h(n) + p_{POP}e_h(n)$	$e_g(n) + e_g(n)$
4	$e_h(n) + p_{POP}e_h(n)$	$e_g(n)$
5	$e_h(n) + p_{RAD}e_h(n)$	$e_g(n) + e_g(n)$
6	$e_h(n) + p_{RAD}e_h(n)$	$e_g(n)$
7	$e_h(n)$	$e_g(n)$
8	0	$e_g(n)$

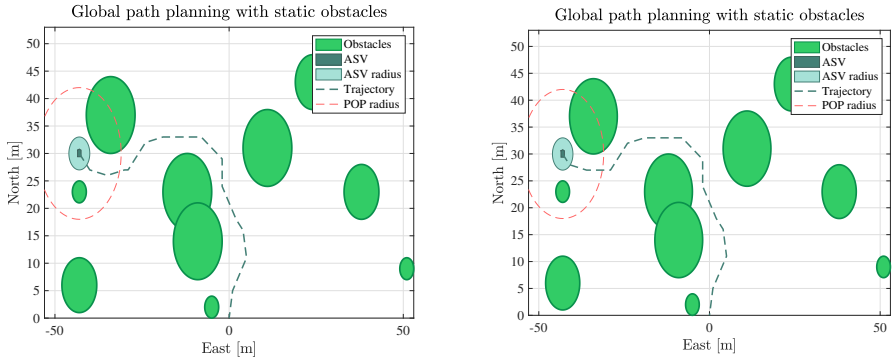
Case 1 and 2

Both trajectories, with $g(n) = 3e_g(n)$ for case 1 and $g(n) = 2e_g(n)$ for case 2, manage to safely avoid the static obstacles in the map. Therefore, based on the the results in figures 6.2a and 6.2b, the scaling of $g(n)$ in case 1 is unnecessary as the $g(n)$ function in case 2 provides equally ideal results. Note that "ideal" refers to path length and risk, where a path that violates the ASV's safety radius is given a larger risk. Further, the vessel's safety radius (r_s) is maintained throughout both paths. However, this does come at a cost of path length, where it is evident that the path planner takes a large turn around the obstacle at $(N, E) \approx (37, -10)$ in NED. However, this turn is a necessity to maintain a safe distance to the obstacles and is a direct result of the feasibility design in section 5.3.1. As both penalty factors, p_{POP} and p_{RAD}, are included in the heuristic, it is not possible to conclude whether it is the POP algorithm or if it is the safety radius that is responsible for the ideal global path. The next cases however, case 3 and 4, sheds some light on this.

Case 3 and 4

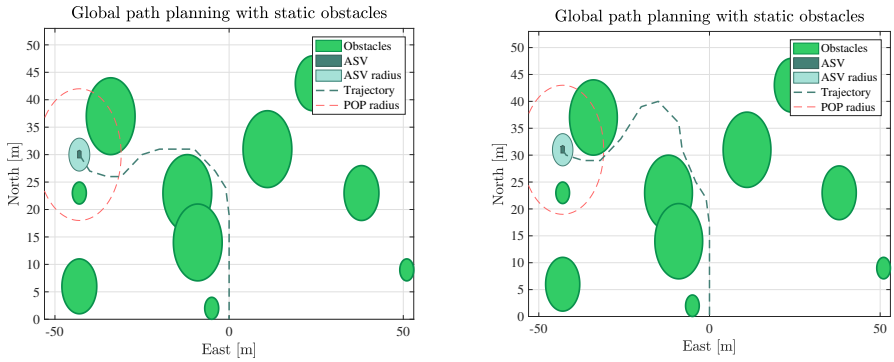
Case 3 and 4 both only use POP to set the heuristic function. As for case 3, with $\beta = 2$ in $g(n)$ from equation (4.7), it is clear that the path planner mainly considers the length of the path. This is the same results that was obtained in case 1, but as the safety radius penalty factor is not included here, the resulting trajectory tends to pass the static obstacles at a shorter distance than in case 1. This, as the A* search now only accounts for the obstacles by placing the nodes containing the obstacles directly in the CLOSED list[1]. Thereby making those nodes inaccessible to the ASV, although their neighbouring nodes are still accessible. This is visible in figure 6.3a. Even though this is an effective approach, it passes the obstacles too close such that the safety of the path is lowered. One solution to overcome this

[1]Refer to section 3.6 for definition of CLOSED list



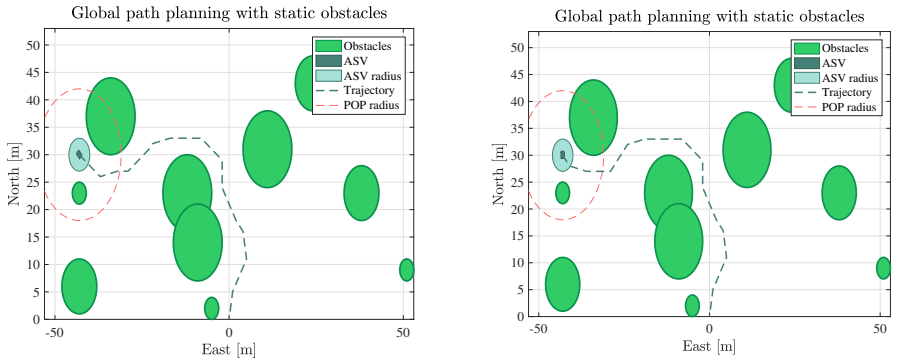
(a) **(b)**

Figure 6.2: The resulting trajectory from the path planner with the total cost function $f(n)$ defined according to (a) case 1 and (b) case 2 in table 6.3.



(a) **(b)**

Figure 6.3: The resulting trajectory from the path planner with the total cost function $f(n)$ defined according to (a) case 3 and (b) case 4 in table 6.3.



(a) **(b)**

Figure 6.4: The resulting trajectory from the path planner with the total cost function $f(n)$ defined according to (a) case 5 and (b) case 6 in table 6.3.

issue, could be to enlarge the static obstacle's representation within the map used for the A* search (see section 5.3.1), however, this solution is too banal and not useful in practice.

For case 4, where $g(n)$ is defined with $\beta = 1$, the resulting path still passes the obstacles too close, thus giving rise to the idea that the POP algorithm handles static obstacles poorly and it is the p_{RAD} penalty factor that is to thank for the safe path in cases 1 and 2. Also, based on figure 6.3b, the resulting path in case 4 is longer than the one obtained in case 3. Therefore, the POP algorithm is inefficient in guiding the A* search towards an optimal and adept global path. The reasons for this inefficiency are investigated further and discussed in chapter 7.

Case 5 and 6

The heuristic functions for cases 5 and 6 rely only on the safety radius penalty factor (p_{RAD}). Much like the previous cases, the two paths for different $g(n)$ functions are almost identical. It is noted that the paths in figure 6.4 are quite like the paths in figure 6.2b, deducing that the contribution from p_{RAD} in $h(n)$ in case 1 affects the A* search to a larger extent than p_{POP}. This explains why the path in case 4 differs from case 2, in that p_{POP} is the deciding factor in case 4 and not in case 2. Seeing as the path in case 2 is favourable compared to case 4, this result is not encouraging for the POP algorithm.

Case 7 and 8

In order to provide the results from cases 1-6 with some reference values, cases 7 and 8 are included. From figure 6.5a it is clear that simply using the Euclidean distance for the cost-to-come and the heuristic results in similar paths to cases 1,

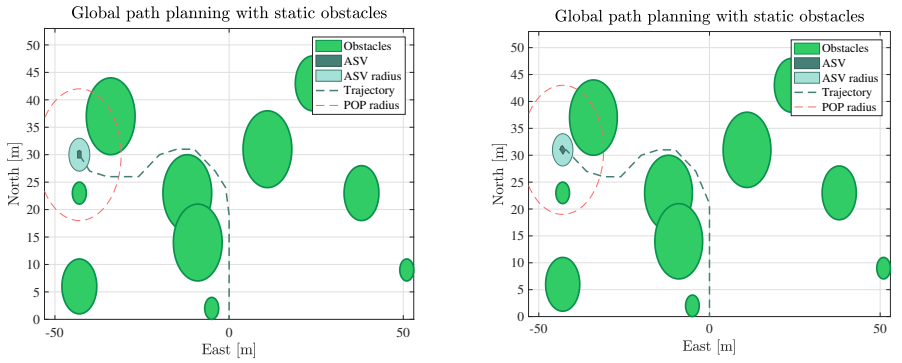


Figure 6.5: The resulting trajectory from the path planner with the total cost function $f(n)$ defined according to (a) case 7 and (b) case 8 in table 6.3.

3 and 5, where $g(n)$ made up the majority of the $f(n)$ value. The path is efficient, although its main flaw is the proximity to the static obstacles. In a real-life scenario, passing static obstacles in this manner can be a highly risky maneuver.

As for case 8, also called Dijkstra's algorithm, the resulting trajectory is quite similar to case 7. The only difference between the two is that case 8 is a marginally shorter path and takes considerably longer computational time to give a resulting path, summarized in table 6.4. As Dijkstra's algorithm is guaranteed to find the shortest path, the path length obtained in case 8 is used as a measure of how the heuristic can decrease computational time but also decrease optimality with regard to path length. This is explained further in the following section.

6.1.3 Summary of Results

In order to compare and contrast the various approaches for the total cost function in cases 1-8, the final path lengths and computational times are summarized in table 6.4.

It is clear from case 8 that Dijkstra's algorithm provides the shortest path, as expected, but with the expense of a long computational time of 2332 seconds. Using the path length in case 8 as a measure for how the heuristic estimates the cost-to-go, it is clear that cases 3 and 7 both have relatively short paths and therefore have accurate heuristic functions. The POP algorithm's path length however, in case 4 in table 6.4, is one of the longest paths. This entails that the cost estimate provided by the the POP algorithm is too inaccurate to provide a proper guide towards the target position for the A* search. The remaining cases all have similar path lengths, where the additional length in cases 1-2 and 5-6 is caused by the penalty factor p_{RAD} that secures a path that passes static obstacles at a distance. As for the computational time, it is clear that the POP algorithm

Table 6.4: Total path lengths and computational times for various design of total cost function $f(n)$ according to table 6.3. These results are for $\mathbf{n_d} = \mathbf{1}$ $[m]$.

Case	Path length [m]	Computational time [s]
1	78.27	305.77
2	77.76	72.52
3	71.20	58.75
4	78.17	12.78
5	77.69	602.89
6	77.76	85.02
7	71.20	67.30
8	71.04	2331.98

heuristic produces a final path in the shortest amount of time. Therefore, if short computational time is favoured over short path length, then the POP algorithm does show some potential. It should however be noted that the trajectory for case 4 in figure 6.3b is too close to the static obstacles to be rendered as safe.

Nevertheless, the POP algorithm is taken a step further and used within the heuristic function in three collision avoidance scenarios. The results are presented in the following section.

6.2 Collision Avoidance

6.2.1 The Scenarios

In a collision avoidance scenario, the ASV is set to avoid the target vessels at a safe distance all the while adhering to COLREGS. After some research, the most common COLREGS scenarios to test are the following:

- Crossing from the right
- Overtaking
- Head-on

Three scenarios with these COLREGS situations were therefore created. In each, the ASV encounters two target vessels (TVs) on its way to the designated waypoint. Each of the target vessels are simulated using a constant velocity model and zero turning rate. These scenarios are summarized in tables 6.5 to 6.7, where the results for each scenario are presented and commented in the following sections. For the remainder of this chapter, TV will be used to refer to the target vessels.

It should be noted that all simulations were completed with a slightly altered POP algorithm than the one outline in section 5.3.4. Rather than using the standard deviation $s = 2$, which had previously been used by [34], the standard deviation was set to $s = 5$ in all collision avoidance maneuvers. It was found that this generated better results with regard to COLREGS, in that the ASV took earlier action to avoid the collision when s was increased to a value of five. More on this in the discussion, in chapter 7.

Table 6.5: Scenario 1: Crossing from the right and head-on

	Initial Position		Waypoints		Speed
Vessel	North [m]	East [m]	North [m]	East [m]	U
ASV	0	0	50	0	1.0
TV 1	16	32	16	-32	2.0
TV 2	55	7	0	7	1.0

Table 6.6: Scenario 2: Head-on and overtaking

	Initial Position		Waypoints		Speed
Vessel	North [m]	East [m]	North [m]	East [m]	U
ASV	5	5	55	25	1.0
TV 1	38	12	1	2	0.82
TV 2	30	21	42	21	0.20

Table 6.7: Scenario 3: Overtaking and crossing from the right

	Initial Position		Waypoints		Speed
Vessel	North [m]	East [m]	North [m]	East [m]	U
ASV	0	0	38	-15	1.0
TV 1	7	-2	21	34	0.60
TV 2	45	7	20	-18	1.41

6.2.2 Scenario 1: Crossing from the Right and Head-On

The first test of the motion planner in a dynamic environment considers TV 1 crossing from the right, followed by TV 2 that approaches the ASV head-on.

In a crossing situation, the avoidance maneuver the ASV must make is to change its course to starboard and pass behind the TV 1, thereby adhering to the COLREGS rules. This action is visible in figure 6.6 from time $t = 0$ to $t = 16$. Based on the ASV's response, it is clear that it follows COLREGS well in a crossing situation. Especially as it starts to make a turn already at $t = 5$, indicating that it *"takes early and substantial action to keep well clear"*[2] of TV 1. The results are therefore encouraging with regard to the procedure for placing the virtual target points (VTPs) in a crossing situation, as described in section 5.4.

Once TV 1 is passed, the ASV must quickly generate a new collision avoidance path to avoid a collision with TV 2. In a head-on situation, COLREGS state that both vessels are to change their course to starboard, such that the vessels pass on the port side of each other. However, as this thesis assumes all other vessels are non-reactive, TV 2 follows its constant course throughout the simulation. As for the ASV, it reacts quickly to the imminent collision, visible in figure 6.6 in $t = 21$, and generates a path that obeys COLREGS and leads to a smooth path onwards to its waypoint.

In both scenarios, the safety radius (r_s) around each vessel is maintained. Although with $r_s = 2$ [m] for all vessels, the ASV passes TV 1 and TV 2 quite close in both collision situations. However, this can be tackled by moving the virtual target points (VTPs) further away from the target vessels, resulting in a more intensive evasive maneuver for the ASV. The implemented tuning parameters used to generate the VTPs for the crossing and head-on situation in this scenario are listed in table 6.8. By, for instance, decreasing γ in a crossing situation, the motion of the VTPs as time passes can be slowed down. Thus, resulting in the ASV changing its course to starboard for a longer period of time than with the current value of γ.

[2]Rule 15 from section 3.2

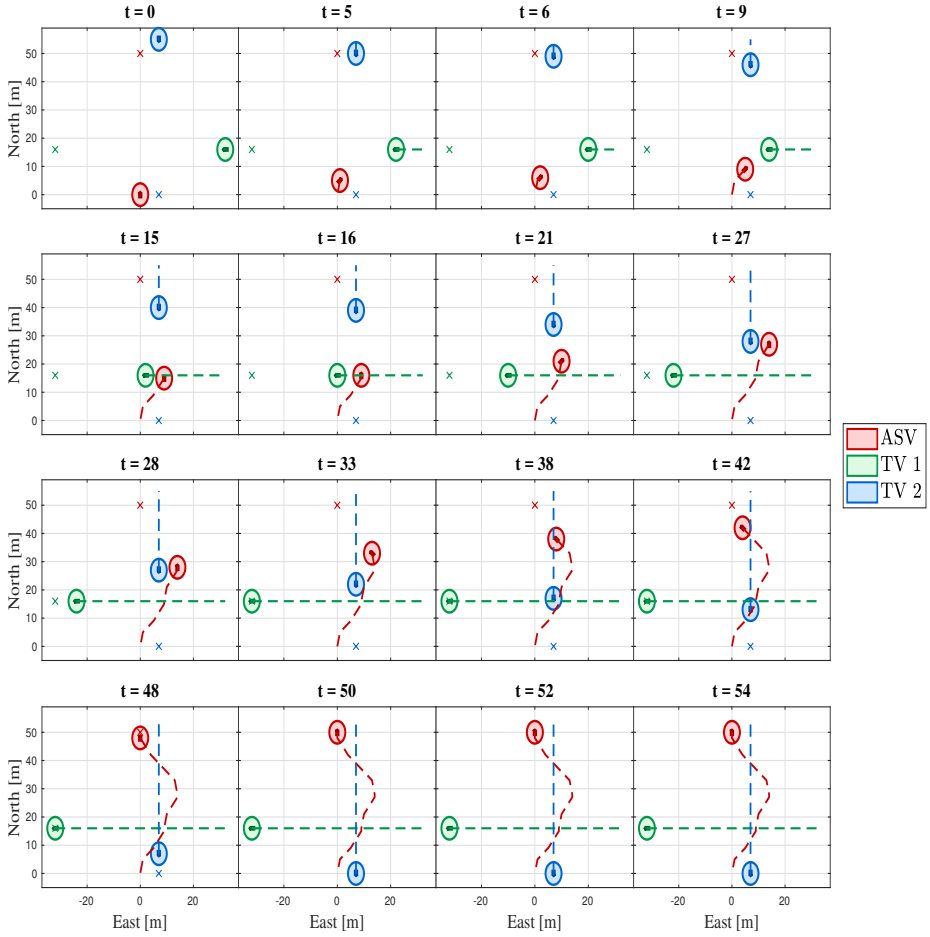


Figure 6.6: Scenario 1: Crossing from the right and head-on. The coloured cross indicates the respective vessel's waypoint, as listed in table 6.5, and t indicates the time of the avoidance maneuver in seconds.

Table 6.8: Scenario 1: Tuning parameters affecting the placement of the virtual target points

	Crossing		Head-on	
Parameters	α	γ	α	κ
Value	0	0.8	1	5

6.2.3 Scenario 2: Head-on and Overtaking

In the second scenario, the ASV is met by TV 1 head-on and must overtake TV 2 to reach its waypoint from table 6.6.

In figure 6.7, the ASV shows that it adjusts its course early to avoid the head-on collision. Repeatedly keeping inline with COLREGS Rule 15, mentioned in scenario 1. The tuning regarding the placement of the VTPs in this head-on situation differ from that in scenario 1, evident by comparing table 6.9 to 6.8. With TV 1 approaching the ASV at an angle, it is necessary to place the VTP further to the right than what was completed in scenario 1. Therefore, κ is increased to $\kappa = 15$ from scenario 1. Using $\kappa = 5$ in this scenario resulted in a less safe and clear avoidance maneuver completed by the ASV. More on this in the discussion, in chapter 7.

In the overtaking situation with TV 2, the ASV has little time to avoid the collision. Based on COLREGS, it can choose which side to pass the overtaking vessel on. In this scenario, the ASV passes TV 2 on the starboard side. From $t = 25$ it generates an evasive maneuver and starts to turn to starboard at $t = 30$. With a T-neighbourhood where the body is equal to five nodes[3] and a constant speed of $U = 1$ [m/s], it is clear that the algorithm chooses to alter its course to starboard in the first stage of the search. This, as the right-most node in the T-neighbourhood is reached after five seconds with these parameters. Therefore, it is apparent that the placement of the VTP in this overtaking situation is well-defined. The tuning parameters are summarized in table 6.9.

Table 6.9: Scenario 2: Tuning parameters affecting the placement of the virtual target points

	Head-on		Overtaking	
Parameters	α	κ	α	κ
Value	1	15	1	10

[3]See table 5.6

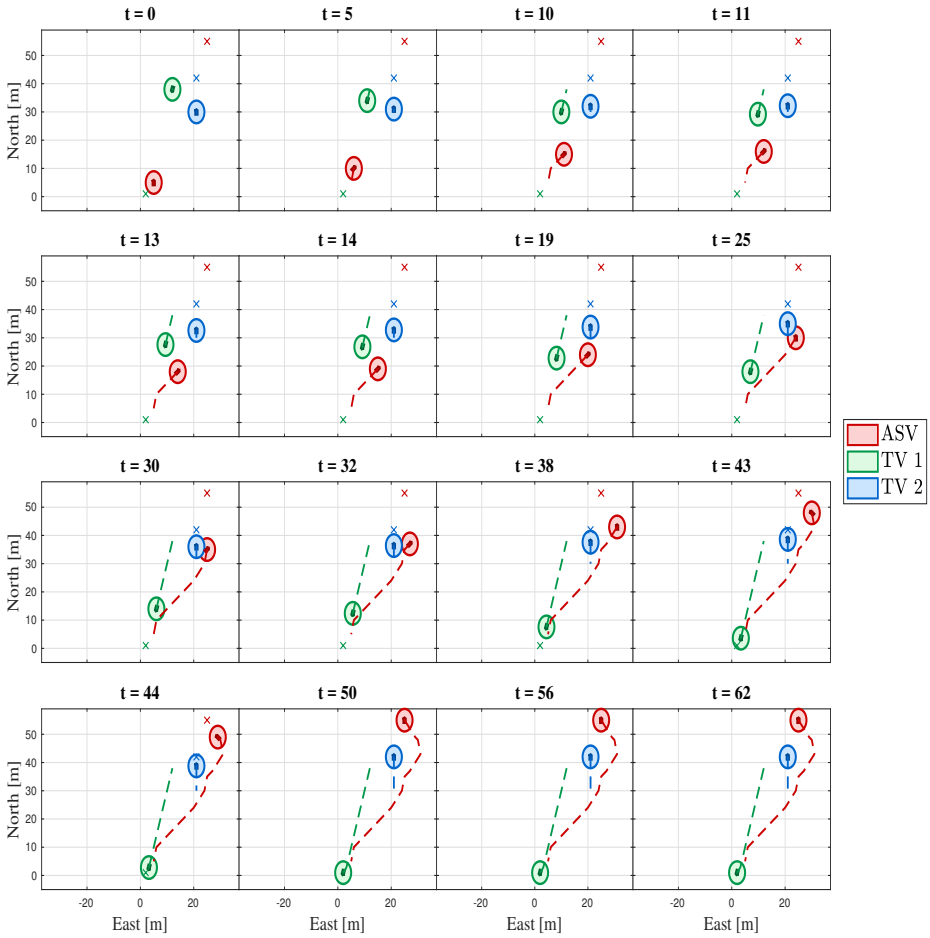


Figure 6.7: Scenario 2: Head-on and overtaking. The coloured cross indicates the respective vessel's waypoint, as listed in table 6.6, and t indicates the time of the avoidance maneuver in seconds.

6.2.4 Scenario 3: Overtaking and Crossing from the Right

In the final collision avoidance scenario, the ASV overtakes one target vessel before it handles a second target vessel approaching from the right. In order to avoid a second collision with TV 1, this vessel makes a turn to the right at $t = 24$ in figure 6.8. This is completed as the collision avoidance algorithm has not been created to handle *multiple* dynamic obstacles, this is left for further work.

The overtaking of TV 1 violates the safety radii around the vessels, evident by $t = 7$ in figure 6.8. Though, it should be noted that the initial positions for the ASV and TV 1 in table 6.7 are positioned close. This is done to test the collision avoidance algorithm in challenging scenarios, and see how fast it can react. Unlike scenario 1 and 2, where the ASV immediately changed its course to starboard once a head-on or overtaking situation was discovered, the ASV uses more time to react in this scenario. In $t = 1$, the A* search has chosen the second node in the body of the T-neighbourhood as the optimal node. It would have been better, with regard to COLREGS and maintaining a safe distance to TV 1, to immediately turn and therefore go to the right-most head node in $t = 5$. This is done in scenario 2, from $t = 25$ to $t = 30$[4]. The reason the motion planner goes to the second body node in $t = 1$, is that all the nodes in the first T-neighbourhood from the initial position have zero probability of reaching the first virtual target point. This limitation is caused by the discretization of the yaw speed, and is further discussed in chapter 7. Nevertheless, the ASV manages to avoid the collision successfully.

The ASV generates an evasive maneuver according to COLREGS in order to avoid TV 2 crossing from the right. Like the overtaking situation with TV 1, the ASV passes TV 2 in a manner such that the safety radii are violated. This is caused by the placement of the VTPs in a crossing from the right situation. As explained in section 5.4, the VTPs are placed at the far-most position of the TV when $\alpha = 0$. However, for a TV approaching at an angle compared the ASV in a crossing situation, as TV 2 does in figure 6.8, this placement is inefficient in securing a safe passing distance between the ASV and TV 2. The VTP placement is further discussed in chapter 7, where its limitations are investigated.

Table 6.10: Scenario 3: Tuning parameters affecting the placement of the virtual target points

	Overtaking		Crossing	
Parameters	α	κ	α	γ
Value	1	15	0	0.8

[4]Refer to figure 5.8 to brush up on the time definitions within each node in the T-neighbourhood

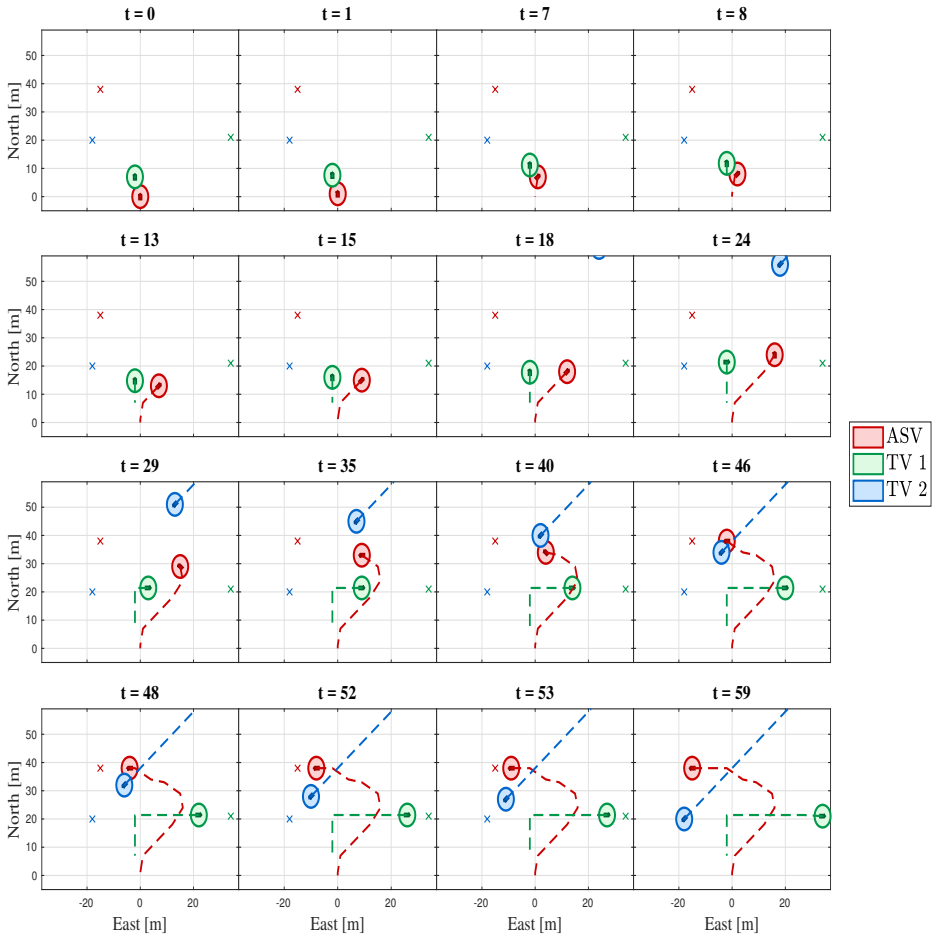


Figure 6.8: Scenario 3: Overtaking and crossing from the right. The coloured cross indicates the respective vessel's waypoint, as listed in table 6.7, and t indicates the time of the avoidance maneuver in seconds.

Chapter 7

Discussion

This chapter presents a further look into the simulation results for the global path planner and the collision avoidance algorithm. Emphasis is placed on investigating the POP algorithm, as the main contribution from this thesis regards the enhancement of POP such that it can handle multiple static obstacles and dynamic obstacles. Therefore, the results from the global path planner that are investigated more closely, includes the POP algorithm in the heuristic design. Further, the pitfalls of using the POP algorithm in collision avoidance are examined in detail.

7.1 Global Path Planning

The global path planner was able to provide a suitable path from the ASV's initial positions to its target position. However, the most optimal paths with regard to computational time and maintaining a safe distance to the static obstacles, were generated without the POP algorithm. All in all, cases 2 and 6 provide the best results in a low computational time. The fact that the A* search with the heuristic function solely dependent on the Euclidean distance and the POP algorithm, case 4, does not result in an equally optimal path is discouraging. In order to comprehend why the A* search leads to produce this sub-optimal path, case 4 is investigated in the discussion below. The remaining cases are not discussed in further depth, as their purpose is to provide a comparison to case 4.

7.1.1 POP Algorithm and the A* search

Using the POP algorithm as the main influence on the heuristic function results in an inadequate path, in that the path in figure 6.3b moves too close to the static obstacles. An interesting comparison to case 4 is case 6, where $h(n)$ is mainly affected by the penalty factor (p_{RAD}) from the safety radius around the ASV.

With this heuristic, the trajectory in figure 6.4b is more optimal with regard to avoiding obstacles. The POP algorithm on the other hand, fails to provide the A* search with accurate information regarding future costs and obstacles, and therefore generates a path that passes the static obstacles too close. As listed in table 6.3, the heuristic function used in this case is $h(n) = e_h(n) + p_{POP}e_h(n)$. Hence, it is clear that using solely information from the POP algorithm results in inefficient paths toward the final position. This section will investigate where the current version of the POP algorithm goes wrong.

POP Parameters

As outlined in section 5.3.4, the parameters used in the POP algorithm are estimates and are therefore subjected to some uncertainty. The effects of altering the standard deviation (s) in the POP algorithm is analyzed; as its value was determined based on separate research in [34]. It is recognized that this research was for a rolling spherical robot, whereas this thesis focuses on developing POP for ASVs. Hence, the standard deviation (s) in the probability density function (PDF) used in POP is tied to doubts. In conjunction with this examination, comes a look at the POP penalty factor (p_{POP}) and how it affects the A* search in the global path planner. First a look at p_{POP}, and how it alters throughout the search.

The POP algorithm's contribution to the global path planner in the form of p_{POP} shows little to no effect in the early stages of the search, a fact made clear by figure 7.1. Plotting p_{POP} from the final path shows how it remains equal to one throughout most of the search. This is equivalent to providing a zero probability of reaching the target node (p) for all nodes in the preliminary stages of the search space, made evident by the plot of p in figure 7.1. In order to investigate how POP is affected by its standard deviation (s), a value which was estimated to $s = 2$ in section 5.3.4, figure 7.1 includes results for both $s = 2$ and $s = 10$. It is interesting to see if altering this estimate, to $s = 10$, changes the results from the A* search in case 4.

The standard deviation (s) affects the probability density function (PDF) in (4.1), which is used to determine the probability of reaching the target node. The effect of increasing the spread in this PDF is visualized in figure 4.3, where a larger value for s entails that the probabilities surrounding the target point have a greater spread throughout the search space. Based on the first plot with $s = 2$, the contribution from the POP algorithm in the form of p_{POP} does not have an effect until the search is ≈ 18 $[m]$ from the target. When the standard deviation is increased, the POP algorithm has an effect much earlier, at ≈ 35 $[m]$ from the target. Noteably, this effect is quite small and converges towards $s = 2$ until the distance to the target position is ≈ 18 $[m]$. Nevertheless, it is speculated that using $s = 10$ will lead to better guidance of the A* search by POP at earlier stages of the search in case 4. However, figure 7.2 shows that that is not the case. The resulting paths are the same for $s = 10$ in figure 7.2a and $s = 100$ in figure 7.2b, where $s = 100$ is included as an extreme value to see how s can affect the search. Comparing

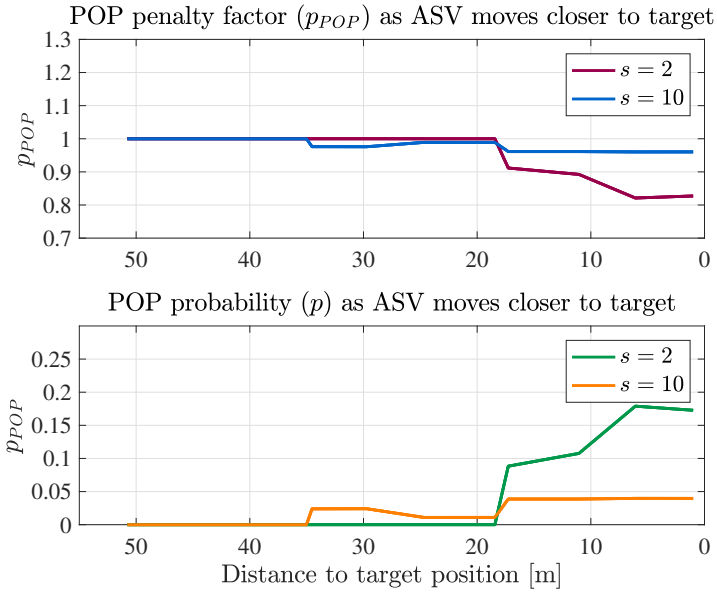


Figure 7.1: The penalty factor from the POP algorithm p_{POP} alters as the search moves close to the target node, the same effect can be seen in the bottom plot with the probability of reaching the target p from the POP algorithm.

these trajectories with figure 6.3b, shows that the trajectory is also the same when $s = 2$. Hence, altering the spread of the probability density function for the POP algorithm does not aid the A* search towards a better global path. The cause for POP's shortcomings in case 4 must therefore originate from a different factor than inaccuracies in the parameter estimates.

Static Obstacles

The discussion above brings forth that it is not the s-value that make the A* search with POP fail compared to case 6. Examining the trajectory in figure 6.3b more closely, reveals how the path starts to turn towards the target point at a much earlier point than the more optimal path generated in case 6. This is shown in figure 7.3, where the plotted waypoints are summarized in table 7.1 along with their respective POP penalty factors. From the figure, it is clear that the motion from point $(N, E) = (22, -1)$ towards $(N, E) = (25, -4)$ causes the path to get too close to the static obstacle and forces the large detour the path experience in case 4. Based on the waypoints in table 7.1, and considering the design of the T-neighbourhood in section 5.3.1, a question as to why the algorithm does not go to $(N, E) = (28, -5)$ where the heading (ψ) would change to $\psi = 0°$ arises. This would have caused a turn away from the static obstacle at an earlier stage than that shown in figure 7.3. However, examining the generated successor nodes

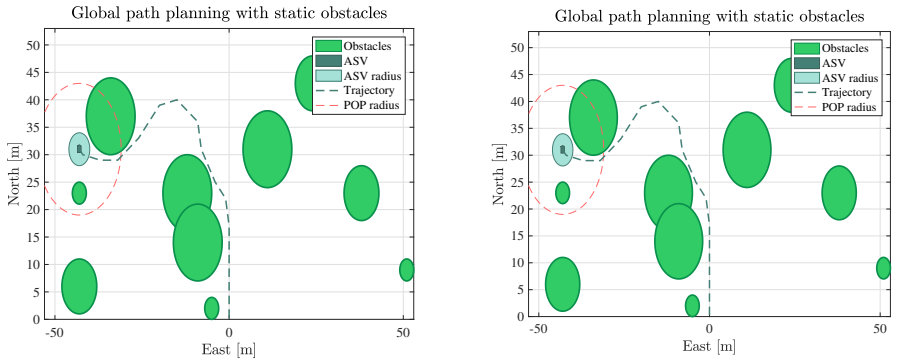


Figure 7.2: Even though the standard deviation in the PDF for reaching the target position is altered from (a) s = 10 to (b) s = 100, the final path from the A* search using the heuristic function $h(n)$ from case 4 remains unaltered.

at the point $(N, E) = (22, -1)$ showed that this choice is justified. The node in $(N, E) = (28, -5)$ is in the CLOSED list in the A* search, which means that it contains an obstacle. The motion from $(N, E) = (22, -1)$ to $(N, E) = (25, -4)$ is visualized in figure 7.4, where the nodes shaded in grey are closed nodes that contain static obstacles.

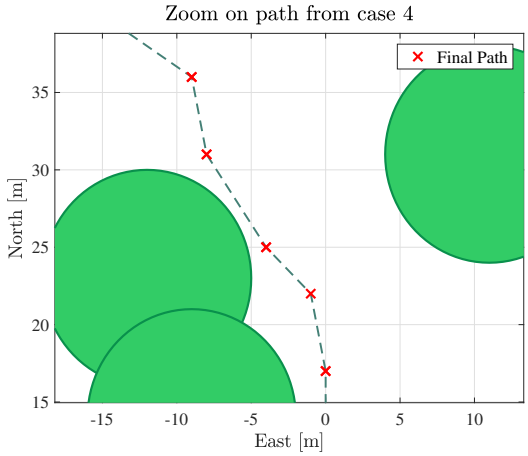


Figure 7.3: The area in case 4 in which the POP algorithm makes an early turn towards the target point but ends up getting too close to the static obstacle.

Table 7.1: Outtake of the final path in case 4, with the path in NED and its respective penalty factors (p_{POP}) listed

North [m]	East [m]	ψ [°]	p_{POP}
17	0	0	1
22	-1	-45	1
25	-4	-45	1
31	-8	0	1
36	-9	-45	0,9897

Looking one step further back, at position $(N, E) = (17, 0)$ with $\psi = 0°$, reveals where the A* search with the POP algorithm makes the mistake that initiates the detour visible in the resulting path in figure 6.3b. Figure 7.5 shows the T-neighbourhood along with the successor nodes from this position. Rather than choosing to go straight ahead, and choose the node shaded in green, the algorithm chooses to go to the blue node in $(N, E) = (22, -1)$ with $\psi = -45°$. It is this choice that traps the path in front of the static obstacle, shown in figure 7.4, and results in the sub-optimal path for case 4.

With this investigation in mind, it can be hypothesized that the fault in the POP algorithm lies in how it handles static obstacles. The heuristic in case 6 results in a path that keeps clear of the obstacles, due to the penalty factor p_{RAD} for violating the safety radius around the ASV, and therefore results in a path without the detour experienced in case 4. The trajectory from POP on the other hand, moves too close to the obstacle. Thus, forcing it to diverge from the optimal path and almost hit a second obstacle on its route towards the target position.

The method for handling static obstacles is collected from [34]. However, this method is based on a *single* static obstacle where the target position lies straight ahead of the current position, as shown in figure 7.6. Once the target position deviates from the current position in a manner such that none of the trajectories generated from solving the SDE[1] are within the PDF for the target position, then all candidate points will have a zero probability of reaching the target. This situation is illustrated in figure 7.7a, where it is clear that using POP in this manner does not mark any of the candidate points as more optimal than the rest. Further, figure 7.7b illustrates how the same situation can occur if the target point is blocked from the ASV's line of sight by too many obstacles. When all successive nodes, i.e the candidate points for the next segment of the path, are given a zero probability of reaching the target, then the search is not given sufficient information to continue towards the target in an optimal manner. Therefore, the turn in figure 7.5 is not caused by the POP algorithm in that it *actively* selects the blue node as the more optimal choice; evident by $p_{POP} = 1$ in table 7.1 for $(N, E) = (22, -1)$, which means that all the successor nodes from $(N, E) = (17, 0)$ were given zero

[1] Step 4 of the POP algorithm on page 50

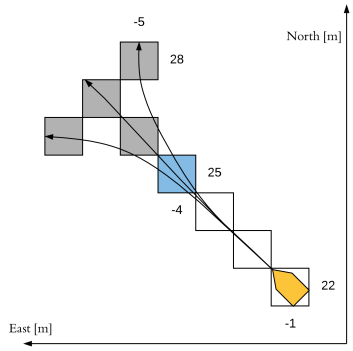


Figure 7.4: The node selection in the T-neighbourhood in case 4 when the ASV is located in $(N, E) = (22, -1)$ with a heading $\psi = -45°$. The best choice with regard to the target point and the static obstacles in the global path planning scenario, would be to move to the grey node located in $(N, E) = (28, -5)$, however this node contains a static obstacle and is therefore unreachable. This is the reason that the choice falls on the node in $(N, E) = (25, -4)$, shaded in blue in this figure.

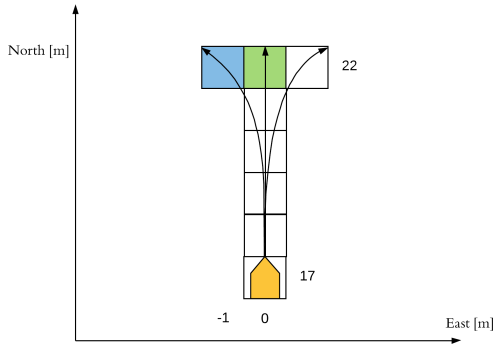


Figure 7.5: The node selection in the T-neighbourhood in case 4 when the ASV is located in $(N, E) = (17, 0)$ with a heading $\psi = 0°$. The best choice with regard to the target point and the static obstacles in the global path planning scenario, would be to move to the green node located in $(N, E) = (22, 0)$. However, due to the aim to keep a short path, the algorithm in case 4 chooses the blue node in $(N, E) = (22, -1)$ with $\psi = -45°$ and starts to move towards one of the static obstacles.

probability of reaching the target. Rather, this turn is caused by the fact that the distance towards the target, i.e $e_h(n)$, is lowest for the node that is closer to the static obstacle. Hence, the turn is not *directly* caused by the POP algorithm, rather it is *indirectly* caused by the POP's lacking ability to determine the node that is most likely to reach the target in a complex environment.

It is therefore established that the way in which POP handles *multiple* static obstacles should be reconsidered. Too often, the nodes are given zero probability of reaching the target, which is also evident in figure 7.1 where p remains equal to zero throughout large parts of the search space. If the A* search is to be properly guided by the POP algorithm, it will need to be able to provide nonzero probabilities for all candidate points in challenging cases, such as the ones presented by figure 7.7.

One possible solution for handling a more complex environment in the global path planner, is to use the same approach that has been developed for dynamic obstacles; where the target points are moved to *virtual* target points (VTPs) that secure that the POP algorithm is effective. Nevertheless, this approach did also suffer from difficulties and is discussed in the following section.

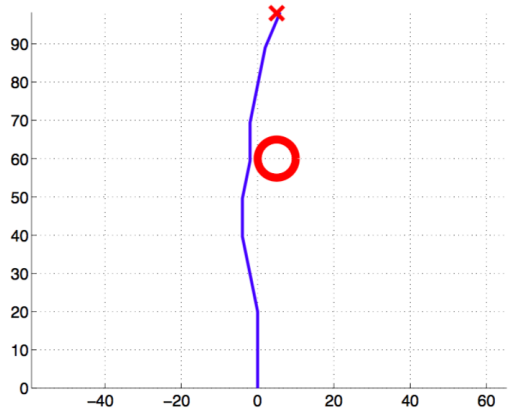
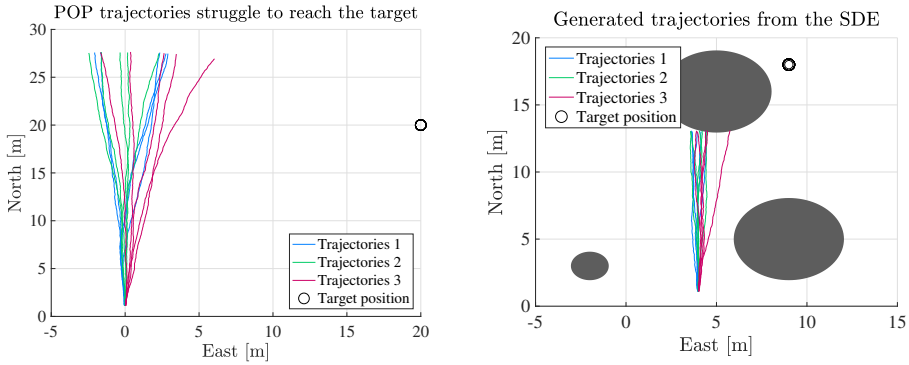


Figure 7.6: The scenario used to demonstrate POP's capability of handling one static obstacle in [34].



(a) **(b)**

Figure 7.7: The trajectories generated from solving the SDEs from three candidate points can give zero probability of reaching the target for all candidate points if (a) the target point is not directly reached by one of the values for the yaw speed, or if (b) too many static obstacles prevent the SDE trajectories to reach the target.

7.2 Collision Avoidance

All in all, the POP algorithm in a collision avoidance scenario outperforms its counterpart in global path planning. Even though the heuristics of the A* search in collision avoidance and global path planning differ, the search is still solely guided by the POP algorithm in case 4 from global path planning and the scenarios for collision avoidance. Based on the discussion for the global path planner, it is clear that the POP algorithm performs well when it has a target point that is within its line of sight. This occurs when the search environment is not cluttered with numerous obstacles, which is not the case for the global path planning scenario in section 6.1, where 10 static obstacles are present. Then a situation like the one depicted in figure 7.7b may occur, where all candidate points have zero probability of reaching the target due to inadequate handling of static obstacles. Contrarily, the approach developed by this thesis for collision avoidance is more tailored towards the POP algorithm. The target point is now placed such that it is within the ASV's line of sight; secured by using virtual target points (VTPs).

The algorithm for placing the VTPs depend on the collision situation at hand, and is constructed to ensure COLREGS compatibility. Based on the results for the three collision avoidance scenarios, it is possible to extend the POP algorithm to handle dynamic obstacles and adhering to COLREGS. However, there are still numerous enhancements that can be made. These are investigated in the following sections. It should also be noted that POP has been merged with an A* search, and is therefore not a complete collision avoidance method on its own.

7.2.1 VTP Placement

As of now, the algorithm for placing the virtual target points (VTPs) for each COLREGS situation is in need of tuning variables, such as γ, κ and α. Even though the results in section 6.2 are promising, the algorithm's dependence on these tuning parameters presents a challenge. If ASVs are to become *truly* autonomous, then they must be able to handle a dynamic environment that is in constant change. This is difficult if the collision avoidance algorithm relies on tuning variables. Therefore, a method that does not have to be tuned according to each collision scenario is preferred. Take for instance the velocity obstacle (VO) approach, examined in the literature review in chapter 2.

The limitation of the tuning variables is evident in scenario 2, where $\kappa = 15$ in the head-on situation, whereas $\kappa = 5$ in the head-on situation in scenario 1. If the same tuning from scenario 1 is used in scenario 2, then the resulting behaviour of the ASV does not adequately follow COLREGS. Based on figure 7.8, the ASV initiates an early course change to starboard in $t = 5$, but alters its course back towards the approaching TV 1 in $t = 13$. This behaviour does not fall in line with COLREGS, where avoidance maneuvers are supposed to be *"large enough to be*

readily apparent to another vessel observing"[2]. Further, say TV 1 is a manned vessel, then the ASV's maneuver in $t = 13$ is certainly set to confuse any human navigator.

The reason for the course change in $t = 13$ is due to inaccurate placement of the VTPs. The current VTP placement algorithm for a head-on situation, outlined in section 5.4, does not incorporate enough parameters to skillfully place the VTPs for *all* sorts of collision scenarios. This counts for overtaking and crossing situations as well. Nevertheless, the focus of this thesis has been to developed the method of the moving VTPs and merge it with the POP algorithm in a collision scenario. Therefore, it was prioritized to make the algorithm work adequately for a set of scenarios. Further work should be to enhance the algorithm by making it handle general collision situations, where tuning of the VTP placements is avoided. Though, the tuning is required at this stage; exemplified by comparing the properly tuned scenario 2 in figure 6.7 versus figure 7.8.

[2]COLREGS Rule 15, see section 3.2

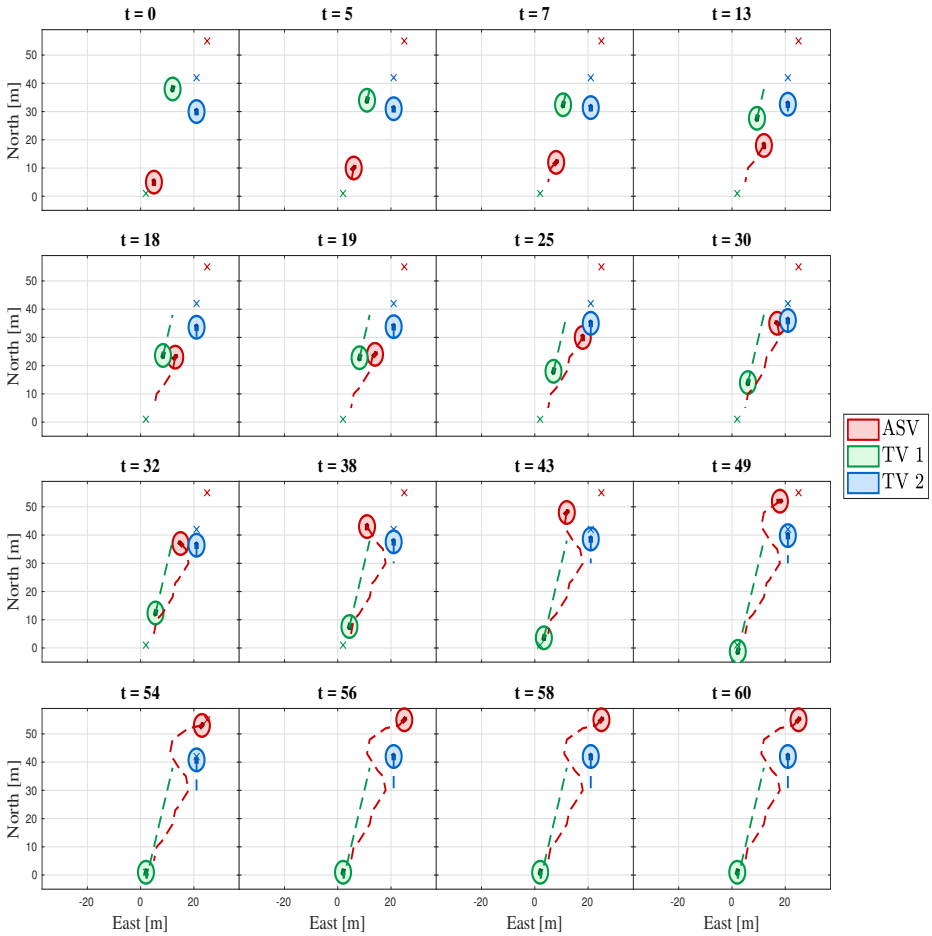
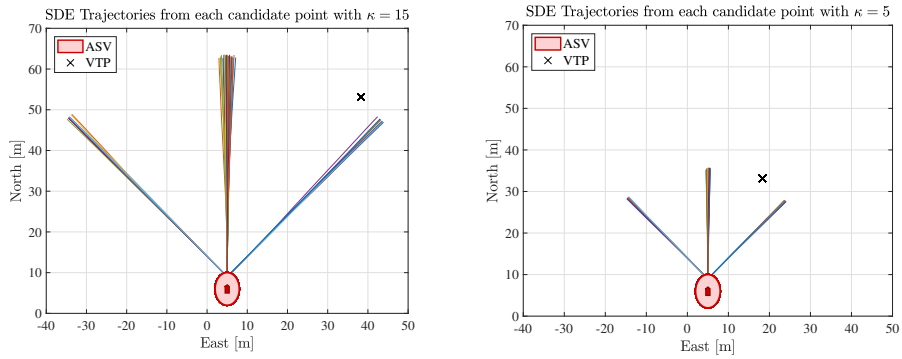


Figure 7.8: The resulting path with $\kappa = 5$ in the head-on situation in scenario 2. The resulting behaviour of the ASV differs from that in figure 6.7.

7.2.2 Limitations caused by the Yaw Speed Discretization

Even though the algorithm for placing the VTPs needs improving, it is put to a challenging task due to the discretization of the yaw speeds. To save memory and allow for a low computational time in the simulations and the A* search, the set of possible yaw speeds is set to $r = [-0.05,\ 0,\ 0.05]$ [rad/s]. This discretization however, raises the demand for accurate positioning of the VTPs.

For instance, figure 7.8 showed that decreasing κ in scenario 2 caused the overall behaviour of the ASV to change for the worse. Figure 7.9 shows why. Using $\kappa = 5$ places the VTP such that it should initiate a course change to starboard. However, with only three yaw speeds to choose from, none of the trajectories manage to hit the VTP in figure 7.9b. The situation is quite similar for $\kappa = 15$ in figure 7.9a, but here it is clear that the first VTP is placed closer to the resulting SDE trajectories for $r = 0.05$ [rad/s]. Therefore, the node that signifies a course change to the right, i.e $r = 0.05$ [rad/s], is set to get a nonzero probability of reaching the target from the POP algorithm with $\kappa = 15$. That same node with $\kappa = 5$ on the other hand, will receive a zero probability of reaching the target, just like the nodes that signify going straight and altering the course to the left, i.e $r = 0$ and $r = -0.05$ [rad/s] respectively. Therefore, the POP algorithm is inefficient in figure 7.9b, giving all the successor nodes a zero probability of reaching the target and thereby failing to guide the A* search in the correct direction. This causes the A* search to simply move to the next node in the OPEN list, which is the node directly ahead of the current position of the ASV. This is what occurs in scenario 3, from $t = 0$ to $t = 1$.



(a) **(b)**

Figure 7.9: The placement of the first VTP with, (a) $\kappa = 15$ and (b) $\kappa = 5$, in the head-on situation in scenario 2.

In order to relax the need for accuracy in the positioning of the VTPs, the standard deviation (s) of the POP algorithm from equation (4.2) is increased. In all the collision avoidance scenarios, $s = 5$, unlike the global path planning scenario where $s = 2$. It has been discussed that increasing s in the global path planning problem yielded no improvements to the overall algorithm, for the collision avoidance cases

however, it did. The effect of increasing the standard deviation meant that the resulting SDE trajectories from $r = 0.05$ [rad/s] in figure 7.9a resulted in a nonzero probability of reaching the VTP. Whereas an $s = 2$, for that very same scenario, resulted in a zero probability of reaching the VTP for $r = 0.05$ [rad/s]. Therefore, the standard deviation of POP was maintained at a larger value for the collision avoidance algorithm.

An improvement to the overall collision avoidance algorithm that includes POP and the A* search, would therefore be to increase the spectrum of the yaw speeds. Increasing the amount of possible yaw speeds would mean that some of the issues regarding the placements of the VTPs could be overcome as the need for accuracy is lowered. This could not be completed on the current simulation platform however, as the memory needed to complete this would be too large to handle. The required amount of memory would drastically increase as the node distance would have to be lowered once the amount of possible yaw speeds increase. This is caused by the feasibility design provided by the T-neighbourhood[3], where each of the nodes in the head represent one yaw speed. Further, each node in the head of the T-neighbourhood is separated by a distance equivalent to the node distance. Therefore, to fit more yaw speeds within the range of $r_{min} = -0.05$ and $r_{max} = 0.05$ [rad/s] into the T-neighbourhood, the node distance would have to decrease correspondingly. This improvement is left for further work.

7.2.3 Handling Dynamic Obstacles

Currently, the method for handling *dynamic* obstacles with the POP algorithm is quite simple. Rather than terminating the SDE trajectories once they are projected to hit a moving obstacle and thereby lowering the probability of reaching the target, which is the method used with static obstacles, the moving obstacles are avoided using the virtual target point (VTP)s.

In the future, the POP algorithm should include the moving obstacles in a more direct manner, such that it is ensured that the nodes that are destined to cause a collision are never chosen. As of now, there is a tendency for the safety radii of the vessels to be violated, as exemplified by scenarios 2 and 3. A solution could be to incorporate a time-dependant CLOSED list within the A* search. This entails placing the nodes that are estimated to contain an obstacle at time t_1 in the CLOSED list at time t_1. Once a time-step has been endured and $t_2 = t_1 + \delta t$, the lists in the A* search are updated such that the nodes that are estimated to contain obstacles at time t_2 are put in the CLOSED list and the CLOSED nodes from t_1 are re-evaluated and possibly moved from the CLOSED list into the OPEN list. This step could not be tested within this thesis, as too much time was spent trying to perfect the global path planner such that little time was left for enhancing the collision avoidance algorithm any further than that presented here. Nevertheless, the adjustments are possible and within range.

[3]Explained in section 5.3.1

Chapter 8

Conclusion

To summarize, a motion planning algorithm for ASVs based on a relatively uncommon approach has been developed throughout the course of this thesis. The Path-of-Probability (POP) algorithm has previously been used for path planning of flexible needles and rolling robots, but not for autonomous surface vessels (ASVs). As for the research question[1], the enhancement made to POP such that it could handle *multiple* static obstacles was brought about by merging POP with the A* algorithm. Thus, the procedure of the POP algorithm was not directly altered. Rather, the use of POP in a cost function was an enhancement which secured that POP could be used in an environment with multiple static obstacles. The set-up of POP was however modified to fit for ASVs. The enhancement of POP for collision avoidance scenarios was completed to a greater extent, in that the virtual target points (VTPs) were used to efficiently overcome the issues from the global path planner. Nevertheless, the procedure of the POP algorithm still remained equal to that within the global path planner[2].

The results for the global path planner brought attention to POP's inefficient handling of multiple static obstacles; POP had a tendency of producing indecisive results in the form of zero probabilities of reaching the target for all successor nodes once the environment was scattered with multiple obstacles. A different approach was used in the collision avoidance module, where the virtual target point (VTP) method was included to ensure decisive results from the POP algorithm. The results were encouraging and showed that POP had the capability of generating COLREGS compliant paths. Nonetheless, the loss of generality that was introduced by the tuning variables in the placement of the VTPs, lowered the credibility of the results. A motion planner must be able to cope with *any* scenario without the need for human intervention, therefore the VTP method should be enhanced such that the need for tuning variables is removed in the future.

[1] See page 2
[2] Refer to section 4.1

Despite some unconvincing results, particularly evident in the global path planner, the POP algorithm should still be subjected to further research regarding motion planning for ASVs. The ocean presents a dynamic and uncertain environment, enabling a need for collision avoidance and global path planning methods that incorporate these stochastic effects into their calculations. As of now, it is the method for handling obstacles that is the main flaw within the algorithm. The VTP method is one suggestion to overcome these flaws, though further research can choose to go in other directions.

Chapter 9

Recommendations for Further Work

9.1 Altering Speed

As of now, the collision avoidance algorithm is generated under the assumption that the ASV maintains a constant forward speed of $U = 1$ [m/s]. This entails that the evasive maneuvers generated by the motion planner is solely based on altering the heading of the vessel. In order to bring another dimension into the motion planner and generate more realistic results, the collision avoidance algorithm should incorporate an altering speed.

Take for instance the avoidance maneuver in the crossing situation in scenario 3, rather than immediately altering its course to starboard, the vessel could also slow down and *then* change its course to starboard. It is speculated that this sort of maneuver would cause the ASV to pass TV 2 with a greater margin than that presented in figure 6.8.

9.2 Model Parameter Estimation

The results presented in chapter 6, all use a version of the POP algorithm with estimated parameters. The results were not perfect and showed that there is room for improvements, discussed in chapter 7. It would therefore be profitable to properly estimate the POP parameters $\boldsymbol{\sigma}$ and s, in order to rule out that it is the *estimates* that are to blame for the POP algorithm's flaws. The values for $\boldsymbol{\sigma}$ and s can be properly assigned with an experimental study of how an ASV behaves in its working environment, thus allowing the stochastic behaviour of the ASV to be studied. The estimation methods from [34] can be used as an inspiration.

9.3 Dynamic Obstacles

9.3.1 Reactive Target Vessels

In the literature review, it was mentioned how most of the current research assumes that the ASV meets target vessels that are *non*-reactive; meaning that the approaching vessels do not complete evasive maneuvers to avoid a prospective collision. This simplification aligns with COLREGS when the ASV is the give-way vessel, such as in a crossing from the right situation. On the other hand, in a head-on situation both vessels are to generate evasive maneuvers, which makes this simplification unrealistic in practice.

Future work regarding motion planning for ASVs should therefore incorporate this added dynamic into the collision avoidance algorithm, such that an accurate portrayal of collision scenarios is used to test the algorithm.

9.3.2 Handling Dynamic Obstacles in POP

The level of the current collision avoidance algorithm using POP can be raised by effectively handling the dynamic obstacles. As of now, it is assumed that the motion and position of each target vessel is perfectly known. However, these measurements are subjected to uncertainties. Therefore, the representation of the dynamic obstacles should mirror the static obstacle representation, where a probabilistic approach is used. This idea is presented in [6], where the probability density function (PDF) used to represent the uncertainty in the target vessel measurements grows as time since the measurement was attained increases.

In the future, the collision avoidance algorithm with POP and an A* search should be enhanced by incorporating the approach outlined by [6]. This can be completed by increasing the costs of the nodes that have a large probability of containing a dynamic obstacle at a given time, thus, the POP algorithm should reduce the probability of reaching the target from that node. The way in which this combination is completed is left for further work.

9.4 Network Complexity

The current set-up of the network used in the A* search, i.e the network that is used to represent the ASV's surrounding environment, is an oversimplification in that only three possible yaw speeds are possible. The ASV can choose from either $r = [-0.05,\ 0,\ 0.05]$ [rad/s]. However, in order to portray a more realistic path choice, this discretization should be altered such that it properly covers the spectrum of the true yaw speeds the ASV can have. This notion is discussed in section 7.2 where the limited motions caused by the current discretization present a challenge to the collision avoidance algorithm.

Future work should therefore include a greater array of possible yaw speeds, and investigate how this can alter the overall behaviour of the ASV with the current collision avoidance algorithm.

Appendix A

Low-Level Controllers

This appendix goes thorough the design of the low-level controllers within the simulator developed in section 5.2. It is included to provide a complete picture of the simulator used to test the motion planning algorithm.

A.1 Heading Controller

The design of the heading controller follows the procedure outlined in [20, p. 374] and therefore starts with a look at the Nomoto model. The Nomoto model is developed from a linearized maneuvering model, as shown in equation (A.1) for the ASV from section 5.2.1, and gives the relationship between the rudder angle (δ) and the yaw rate (r). Note that $\boldsymbol{\nu} = [v,\ r]^T$ in equation (A.1). From the actuator models in section 5.2.2, the $\boldsymbol{b}$-matrix in equation (A.1) is given by the expression in equation (A.2). Note that the surge speed u, is kept at a constant value equal to u_0.

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{b}\delta \tag{A.1}$$

$$\boldsymbol{b} = \begin{bmatrix} F_y \\ M_\psi \end{bmatrix} \begin{bmatrix} K_{rud}u_0^2 \\ l_x K_{rud}u_0^2 \end{bmatrix} \tag{A.2}$$

As the heading controller aims to control the yaw rate, r, the Nomoto model is obtained by investigating the dynamic equation for r from equation (A.1) and then transforming the resulting equation to the Laplace plane. This results in two possible Nomoto models, one of first-order and one of second-order. These are shown by equations (A.3) and (A.4) respectively, where the first-order Nomoto model is obtained by setting $T := T_1 + T_2 - T_3$ in the second-order Nomoto model.

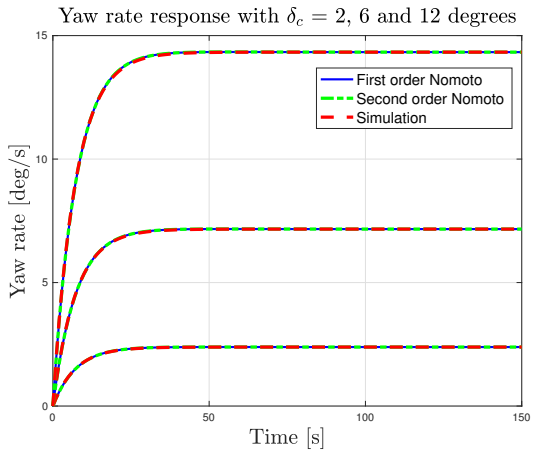


Figure A.1: The yaw rate response of the implemented ASV model after step-inputs in the rudder command. The yaw rate response for the same rudder commands are also shown for the first- and second-order Nomoto models.

Table A.1: The resulting coefficients for the first- and second-order Nomoto models after curve-fitting in MATLAB.

δ [deg]	K	T	T_1	T_2	T_3
2	1.3040	7.2406	7.5029	7.5102	7.5405
6	1.3040	7.2406	15.7921	15.5064	14.3291
12	1.3040	7.2406	16.1447	15.8654	14.7145

$$\frac{r}{\delta}(s) = \frac{K}{(1+Ts)} \tag{A.3}$$

$$\frac{r}{\delta}(s) = \frac{K(1+T_3 s)}{(1+T_1 s)(1+T_2 s)} \tag{A.4}$$

Now, in order to determine which of the two Nomoto models that are best suited for the ASV model presented in section 5.2.1, the ASV's yaw rate response as a function of rudder input is compared to the yaw rate response of the two Nomoto models. The coefficients in equations (A.3) and (A.4) are determined using a least-square curve-fit to the ASV's simulated yaw rate time-series. The resulting yaw rates are shown in figure A.1 and the coefficients from the curve-fitting are presented in table A.1.

From figure A.1 it is clear that the first-order Nomoto model can be used to represent the vessel dynamics of the modelled ASV. Further, as each of the coefficients for K and T in table A.1 are equal for all step-inputs in the rudder command

(which makes sense as the actual model is linear), these values are used in the development of the heading controller.

With a linear ASV model, it is not necessary to create an overly complicated heading controller. Therefore, the standard proportional-integral-derivative (PID) controller is used. The controller output is given by equation (A.5), where $\tilde{\psi} := \psi - \psi_d$ and $\tilde{r} = \dot{\tilde{\psi}} := \dot{\psi} - \dot{\psi_d}$.

$$\delta_c = -K_p\tilde{\psi} - K_d\tilde{r} - K_i \int_0^t \tilde{\psi}(\tau)d\tau \tag{A.5}$$

The controller gains are determined by following the pole-placement algorithm in [20, p. 374]. The first step is to set the bandwidth of the controller (ω_n) and the desired relative damping ratio (ζ). As the needed bandwidth of the heading controller will depend upon the trajectory generated by the guidance system, its value was set quite high; ensuring that the controller can follow most input signals. Specifically $\omega_b = 0.15$ [rad/s] and $\zeta = 1$ to secure a smooth response. The gains are then determined by equations (A.7)-(A.9), where the natural frequency of the closed-loop system is determined using ω_b and ζ in equation (A.6).

$$\omega_n = \frac{\omega_b}{\sqrt{1 - 2\zeta^2 + \sqrt{4\zeta^4 - 4\zeta^2 + 2}}} \tag{A.6}$$

$$K_p = \frac{\omega_n^2 T}{K} \tag{A.7}$$

$$K_i = \frac{\omega_n^3 T}{10K} \tag{A.8}$$

$$K_d = \frac{2\zeta\omega_n T - 1}{K} \tag{A.9}$$

Using the equations above and the coefficients from table A.1, the gains for the ASV model were calculated and implemented within the simulator. The controller was tested for various input signals, one sinus-curve with a frequency of 0.01 [rad/s] and amplitude of 20 [°]; and one step-input with an amplitude of 20 [°]. It was found that reducing the Nomoto gain K produced more responsive results when it came to the sine-input, but the calculated K value worked well with a step-input. Therefore, two sets of gains were used, where the nature of the input signal decided which of the two sets of gains were used. The resulting controller values are given in table 5.3, where the K value from table A.1 was used to calculate the gain values as the nature of the LOS laws is more similar to a step-input than a sine-input. The heading response and commanded rudder angle for each of the two input signals are shown in figures A.2 and A.3, where the saturation level of the rudder and propeller shaft speed are set to $\delta_{max} = \pm 15$ [°] and $n_{max} = \pm 8.9[rad7s]$ respectively.

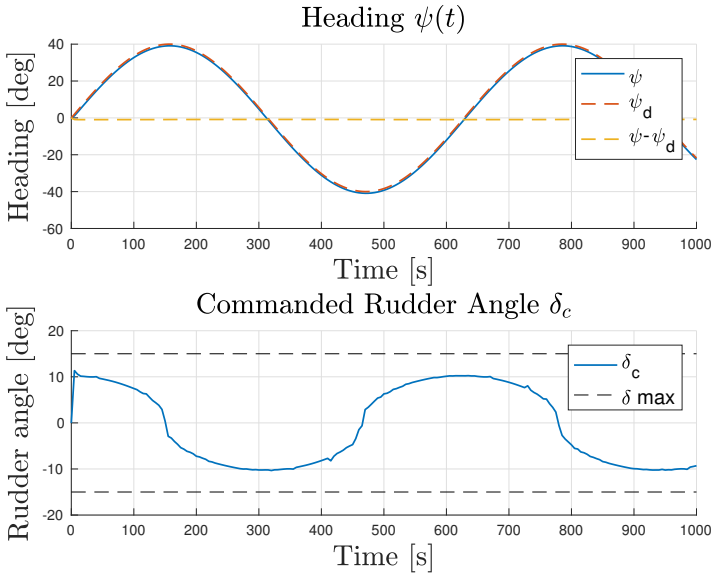


Figure A.2: The heading response of the ASV model implemented in the simulator with a PID controller, here following a sine-input in the desired heading angle ψ_d.

A.2 Surge Speed Controller

Much like the heading controller, the surge speed controller is implemented using the linearized maneuvering equation. However, as the ASV model is already linear and decoupled in a surge and sway-yaw sub-system, this is not technically needed. Nevertheless, neglecting environmental disturbances gives the surge model in equation (A.10), where m_{11} and d_{11} refers to the mass- and damping-matrix values in equations (5.1)-(5.2). As the ASV only has one propeller located at the stern, the resulting force in the surge direction τ_1, is given according to equation (A.11); where n is the shaft speed in rad/s. The propeller coefficient $K_{prop} = 2$, by recommendation of my supervisor.

$$m_{11}\dot{u} - d_{11}u = \tau_1 \tag{A.10}$$
$$\tau_1 = K_{prop} n|n| \tag{A.11}$$

With a linear system in equation (A.10), the choice for a controller falls on a simple PI controller; neglecting the derivative term to avoid derivating the surge speed measurements. The derivative term could easily be implemented, say if a navigation system were to be implemented to filter the measurements, but as the PI controller proved to worked well with little tuning, there is no need for

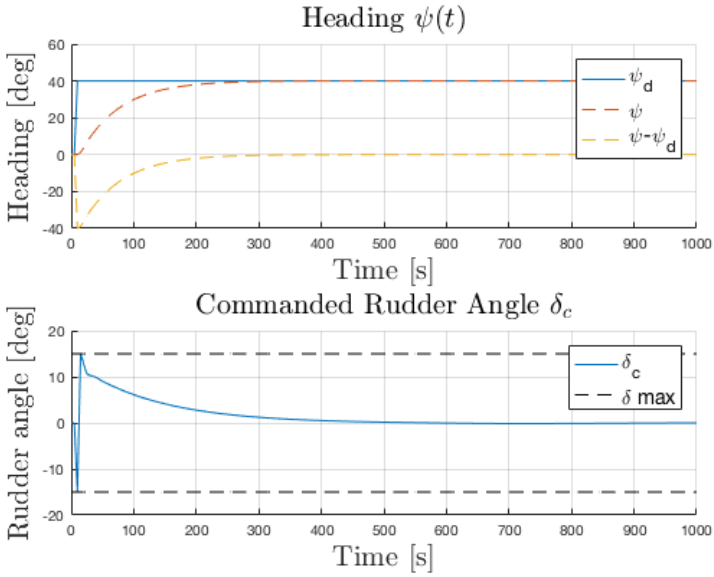


Figure A.3: The heading response of the ASV model implemented in the simulator with a PID controller, here following a step-input in the desired heading angle ψ_d.

it. The resulting output from the PI controller is given in equation (A.12), where $\tilde{u} := u - u_d$ and the controller gains were tuned manually according to equation (A.13). The resulting behaviour of the model ASV is shown in figure A.4, where the quick response of the vessel is evident and the fact that the shaft speed avoids saturation is a plus.

$$n_c = -K_p\tilde{u} - K_i \int_0^t \tilde{u}(\tau)d\tau \tag{A.12}$$

$$K_p = 1 \qquad K_i = \frac{K_p}{10} = 0.1 \tag{A.13}$$

It should be noted that the propeller force in equation (A.11) makes the closed-loop system nonlinear when the surge controller outputs a shaft speed n_c, shown by equation (A.14). However, as the main contribution from this thesis regards the guidance system, simplicity is a key factor in the design of the low-level controllers. Therefore, the PI controller in equations (A.12) and (A.13) remains implemented in the simulator.

$$\frac{m_{11}}{K_{prop}}\dot{u} - \frac{d_{11}}{K_{prop}}u = n_c|n_c| \tag{A.14}$$

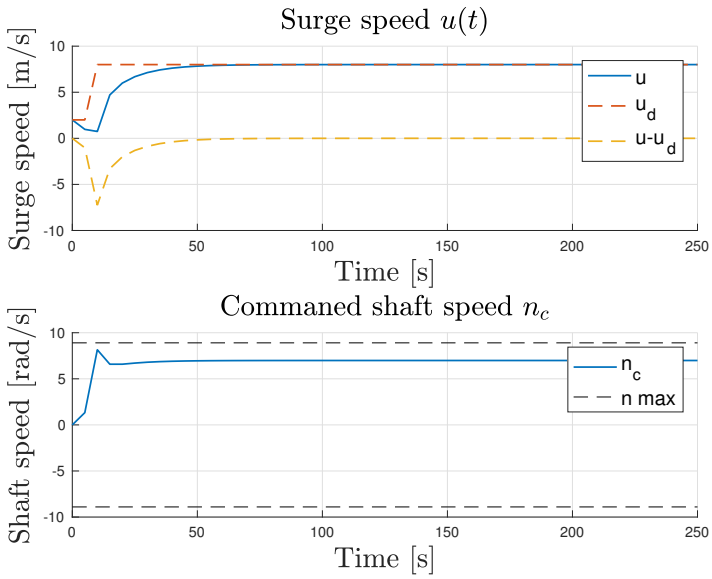


Figure A.4: The ASV model's surge response following the implementation of a surge speed controller. Here, the desired surge speed is set to $u_d = 8$ [m/s] and the saturation level of the shaft speed is set to $n_{max} = 8.9$ [rad/s].

Appendix B

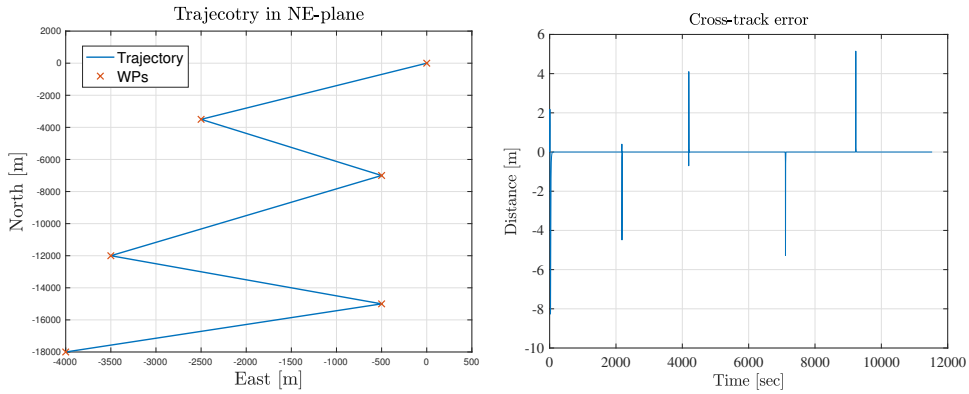
Guidance Controller

In order to show the capability of the guidance controller and its parameters discussed in section 5.2.4, a run with a set of waypoints given by table B.1 is shown in this chapter.

Table B.1: Waypoints used to the test the guidance controller

North [m]	East [m]
0	0
-3500	-2500
-7000	-500
-12000	-3500
-15000	-500
-18000	-4000

The resulting behaviour of the ASV is shown in figure B.1, where figure B.1a shows how the guidance controller keeps the ASV model on the correct course. Further, figure B.1b shows how the cross-track error $e(t)$ is quickly reduced to zero, and has peaks once the guidance scheme switches to the following waypoint - which is to be expected. Hence, the guidance controller proves to be effective.



(a) **(b)**

Figure B.1: (a) The guidance controller manages to follow the set of waypoints. (b) Cross-track error $e(t)$ during the track following.

Bibliography

[1] *A Brief History of Autonomous Vehicle Technology.* https://www.wired.com/brandlab/2016/03/a-brief-history-of-autonomous-vehicle-technology/. Accessed online. 2018.

[2] Pranay Agrawal and John M Dolan. "COLREGS-compliant target following for an Unmanned Surface Vehicle in dynamic environments". In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on.* IEEE. 2015, pp. 1065–1070.

[3] Ron Alterovitz, Michael Branicky, and Ken Goldberg. "Motion Planning Under Uncertainty for Image-guided Medical Needle Steering". In: *The International Journal of Robotics Research* 27.11-12 (Nov. 2008), pp. 1361–1374. ISSN: 0278-3649.

[4] *Autonomous ship project, key facts about YARA Birkeland.* https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/?OpenDocument. Accessed online. 2017.

[5] Michael Blaich et al. "Fast grid based collision avoidance for vessels using $A*$ search algorithm". In: *Methods and Models in Automation and Robotics (MMAR), 2012 17th International Conference on.* IEEE. 2012, pp. 385–390.

[6] Michael Blaich et al. "Probabilistic Collision Avoidance for Vessels". English. In: *International Federation of Automatic Control (IFAC)* 48.16 (2015), pp. 69–74. ISSN: 2405-8963.

[7] Ilhem Boussaid, Julien Lepagnot, and Patrick Siarry. "A survey on optimization metaheuristics". In: *Information Sciences* 237.03 (Feb. 2012), pp. 82–117.

[8] Robert Brown and Patrick Hwang. *Introduction to Random Signals and Applied Kalman Filtering.* English. 3rd Edition. John Wiley & Sons, 1997. ISBN: 0-471-12839-2.

[9] Eduardo F Camacho. *Model Predictive Control.* eng. 2nd ed. Advanced Textbooks in Control and Signal Processing. 2007. ISBN: 0-85729-398-2.

[10] Sable Campbell and Wasif Naeem. "A rule-based heuristic method for COLREGs-compliant collision avoidance for an unmanned surface vehicle". In: *IFAC Proceedings Volumes* 45.27 (2012), pp. 386–391.

[11] Giuseppe Casalino, Alessio Turetta, and Enrico Simetti. "A three-layered architecture for real time path planning and obstacle avoidance for surveil-

lance USVs operating in harbour fields". In: *Oceans 2009-Europe*. IEEE. 2009, pp. 1–8.

[12] Cherry Cheung. "Selective Velocity Obstacle method for Autonomous Collision Avoidance: An experimental validation". Master's thesis. Delft University of Technology, 2016.

[13] Rodney Coleman. *Stochastic processes*. Vol. 14. Springer Science & Business Media, 2013.

[14] James Colito. "Autonomous Mission Planning and Execution for Unmanned Surface Vehicles in Compliance with the Marine Rules of the Road". Master's thesis. University of Washington, 2007.

[15] *Convention on the International Regulations for Preventing Collisions at Sea, 1972 (COLREGs)*. `http://www.imo.org/en/About/Conventions/ListOfConventions/Pages/COLREG.aspx`. Accessed online. 2018.

[16] Imme Ebert-Uphoff and Gregory S. Chirikjian. "Inverse Kinematics of Discretely Actuated Hyper-Redundant Manipulators using Workspace Densities". In: *IEEE International Conference on Robotics and Automation* 1 (1996), pp. 139–145. ISSN: 10504729.

[17] Maren Kristine Eidal. *A Literature Review: How to Solve the Path Planning Problem for Autonomous Surface Vessels*. Project Thesis at NTNU for Marine Technology. Dec. 2017.

[18] Jose B. Escario, Juan F. Jimenez, and Jose M. Giron-Sierra. "Optimisation of autonomous ship manoeuvres applying Ant Colony Optimisation metaheuristic". eng. In: *Expert Systems With Applications* 39.11 (Sept. 2012), pp. 10120–10139. ISSN: 0957-4174.

[19] Paolo Fiorini and Zvi Shiller. "Motion planning in dynamic environments using velocity obstacles". In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772.

[20] Thor I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. English. John Wiley & Sons, 2011. ISBN: 1-283-40556-3.

[21] *Genetic Algorithms and Evolutionary Algorithms - Introduction*. `https://www.solver.com/genetic-evolutionary-introduction`. Accessed online. 2017.

[22] Inger Berge Hagen. "Collision Avoidance for ASVs Using Model Predictive Control". Master's thesis. Norwegian University of Science and Technology, 2017.

[23] *Heuristics*. `http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html`. Accessed online. 2009.

[24] Desmond J. Higham. "An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations". English. In: *SIAM Review* 43.3 (2001), pp. 525–546. ISSN: 0036-1445.

[25] International Maritime Organization (IMO). *COLREGS - International Regulations for Preventing Collisions at Sea*. Version 9.4. 1972.

[26] *Introduction to A**. `http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html#the-a-star-algorithm`. Accessed online. 2009.

[27] Tor Arne Johansen, Tristan Perez, and Andrea Cristofaro. "Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment". In: *IEEE Transactions on Intelligent Transportation Systems* 17.12 (Dec. 2016), pp. 3407–3422.

[28] Ioannis Karatzas and Steven Shreve. *Brownian Motion and Stochastic Calculus*. English. 2nd Edition. Vol. 113. Graduate Texts in Mathematics. 1998. ISBN: 1-4612-0949-8.

[29] Andrew Kimmel, Athanasios Krontiris, and Kostas Bekris. *Decentralized Motion Coordination and Team Coherence*. `http://www.pracsyslab.org/motion_coordination`.

[30] B. Kluge and E. Prassler. "Recursive probabilistic velocity obstacles for reflective navigation". In: *Springer Tracts in Advanced Robotics* 24 (2006), pp. 71–79. ISSN: 16107438.

[31] Yoshiaki Kuwata et al. "Safe Maritime Autonomous Navigation With COLREGS, Using Velocity Obstacles". English. In: *IEEE Journal of Oceanic Engineering* 39.1 (Jan. 2014), pp. 110–119. ISSN: 0364-9059.

[32] Agnieszka Lazarowska. "Swarm Intelligence Approach to Safe Ship Control". In: *Polish Maritime Research* 22 (2015), pp. 34–40.

[33] Agnieszka Lazarowska. "Multi-criteria ACO-based Algorithm for Ship's Trajectory Planning". In: *The International Journal on Marine Navigation and Safety of Sea Transportation* 11.1 (Mar. 2017).

[34] Jaeyeon Lee and Wooram Park. "Design and path planning for a spherical rolling robot". In: *ASME 2013 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers. 2013.

[35] Jaeyeon Lee and Wooram Park. "A probability-based path planning method using fuzzy logic". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE. 2014, pp. 2978–2984.

[36] Anastasios M Lekkas and Thor I Fossen. "A time-varying lookahead distance guidance law for path following". In: *IFAC Proceedings Volumes* 45.27 (2012), pp. 398–403.

[37] Hoi-Shan Lin, Jing Xiao, and Zbigniew Michalewicz. "Evolutionary algorithm for path planning in mobile robot environment". In: *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. IEEE. 1994, pp. 211–216.

[38] Jan Lundgren, Mikael Ronnqvist, and Peter Varbrand. *Optimization*. English. Studentlitteratur, 2010. ISBN: 9789144053080.

[39] Signe Moe. "Guidance and control of robot manipulators and autonomous marine robots". eng. PhD thesis. Trondheim, 2016. ISBN: 978-82-326-1982-5.

[40] Signe Moe and Kristin Ytterstad Pettersen. "Set-Based Line-of-Sight (LOS) Path Following with Collision Avoidance for Underactuated Unmanned Surface Vessel". eng. In: IEEE, 2016. URL: `http://hdl.handle.net/11250/2434096`.

[41] Signe Moe et al. "Experimental Results for Set-based Control within theSingularity-robust Multiple Task-priority Inverse KinematicsFramework". eng. In: IEEE, 2015. URL: `http://hdl.handle.net/11250/2393340`.

[42] Helene Myre. "Collision Avoidance for Autonomous Surface Vehicles Using Velocity Obstacle and Set-Based Guidance". Master's thesis. Norwegian University of Science and Technology, 2016.

[43] *New test-bed for autonomous ships opens in Norway.* `https://safety4sea.com/new-test-bed-for-autonomous-ships-opens-in-norway/`. Accessed online. Dec. 2017.

[44] Wooram Park, Yunfeng Wang, and Gregory S Chirikjian. "The path-of-probability algorithm for steering and feedback control of flexible needles". In: *The International journal of robotics research* 29.7 (2010), pp. 813–830.

[45] Wooram Park et al. "Estimation of model parameters for steerable needles". English. In: IEEE Publishing, May 2010, pp. 3703–3708. ISBN: 978-1-4244-5038-1.

[46] Nicolas Privault. *Understanding Markov Chains : Examples and Applications.* English. Springer Undergraduate Mathematics Series. 2013. ISBN: 981-4451-51-7.

[47] Taehyun Shim, Ganesh Adireddy, and Hongliang Yuan. "Autonomous vehicle collision avoidance system using path planning and model-predictive-control-based active front steering and wheel torque control". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 226.6 (June 2012), pp. 767–778. ISSN: 0954-4070.

[48] Roman Smierzchalski. "Trajectory planning for ship in collision situations at sea by evolutionary computation". In: *IFAC Proceedings Volumes* 30.22 (1997), pp. 129–134.

[49] Øyvind Notland Smogeli. "Control of marine propellers: from normal to extreme conditions". English. PhD thesis. Trondheim, 2006.

[50] Sverre Steen. *TMR4220 Naval hydrodynamics foil and porpeller theory : lecture notes.* English. Trondheim, 2014.

[51] Thomas Stenersen. "Guidance System for Autonomous Surface Vehicles". Master's thesis. Norwegian University of Science and Technology, 2015.

[52] *The ReVolt: A new inspirational ship concept.* `https://www.dnvgl.com/technology-innovation/revolt/index.html`. Accessed online. 2013.

[53] Ming-Cheng Tsou and Chao-Kuang Hsueh. "The study of ship collision avoidance route planning by ant colony algorithm". In: *Journal of Marine Science and Technology* 18.5 (2010), pp. 746–756.

[54] Ming-Cheng Tsou, Sheng-Long Kao, and Chien-Min Su. "Decision support from genetic algorithms for ship collision avoidance route planning and alerts". In: *The Journal of Navigation* 63.1 (2010), pp. 167–182.

[55] Ronald Walpole et al. *Probability & statistics for engineers & scientists.* English. 9th Edition. Boston, Mass: Pearson, 2012. ISBN: 9780321748232.

[56] Qingyang Xu, Chuang Zhang, and Ning Wang. "Multiobjective optimization based vessel collision avoidance strategy optimization.(Research Article)". eng. In: *Mathematical Problems in Engineering* (Jan. 2014). ISSN: 1024-123X.

[57] Lotfi A. Zadeh. "Fuzzy Logic, Neural Networks, and Soft Computing". In: *Communications of the ACM* 37.03 (Mar. 1994), pp. 77–84.