Master's thesis

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

Torstein Fagerlund

Sample Based Model Predictive Control for Path Following and Collision Avoidance of Unmanned Surface Vehicles

Master's thesis in Industrial Cybernetics

Supervisor: Kristin Y. Pettersen

June 2020

Courtesy of FFI.

NTNU
Norwegian University of Science and Technology

FFI Forsvarets forskningsinstitutt
Norwegian Defence Research Establishment

Torstein Fagerlund

Sample Based Model Predictive Control for Path Following and Collision Avoidance of Unmanned Surface Vehicles

Master's thesis in Industrial Cybernetics
Supervisor: Kristin Y. Pettersen
June 2020

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

NTNU
Norwegian University of
Science and Technology

Problem description

The Norwegian Defence Research Establishment (Forsvarets forskningsinstitutt, FFI) is currently investigating the use of unmanned surface vehicles (USVs) for use in various military applications. Two test platforms, Odin and Frigg, have been developed to investigate the use of this technology in mine countermeasures operations and as support vessels for autonomous underwater vehicles. The vehicles will operate autonomously, which requires a high degree of robustness and safety in the control algorithms.

The vehicles will be required to operate safely and efficiently in complex dynamic areas. The vehicles will have multiple goals, which may in part be conflicting, as is the case with path following and collision avoidance. Furthermore, even with no conflicting goals the scenarios can be complex, e.g. when moving through an area cluttered with obstacles. In these situations, it is advantageous for the vehicle to be able to plan its action ahead, increasing the likelihood of optimal vehicle trajectories.

Such planning algorithms can be computationally expensive, making them infeasible for real-time obstacle avoidance. Furthermore, the complexity associated with path following and collision avoidance makes the set of feasible solution highly non-convex, making it difficult for traditional algorithms to find a global optimum. In recent years, however, the use of sampling-based model predictive control (MPC) has increased in popularity for maritime use. Sampling-based MPC avoids the cost of a gradient search by simply searching among a limited set of possible trajectories. By choosing the optimal trajectory among the candidates, a global solution is found (albeit in a limited search space).

This thesis will investigate the use of this family of algorithms. Specifically, the suitability of sample-based MPC in areas where precise control are required is of interest, as this poses a challenge when the solution space is heavily subsampled. Such scenarios include path following and collision avoidance in cluttered environments.

Specifically, the thesis will investigate the use of sampling-based MPC to make an underactuated USV such as Odin and Frigg

- Minimizing the distance to a path or trajectory
- Keeping at least a minimum distance to other vessels and obstacles
- Following the international regulations for avoiding collisions at sea (COLREGs)
- Retain predictability by avoiding (too) frequent course adjustments

Proposed subtasks for this thesis are:

- Perform a literature study in the field of path- and motion-planning for unmanned vehicles, with particular focus on underactuated vehicles or vehicles with non-holonomic constraints
- Compare different approaches to sampling-based MPC
- Implement at least one sampling-based MPC algorithm in a simulated environment
- Investigate the use of the implemented sampling algorithm(s) for path/trajectory following and collision avoidance

The thesis will build upon preliminary work performed in a previous project assignment.

Abstract

The Norwegian Defence Research Establishment (Forsvarets forskningsinstitutt, FFI) has for several years been researching subsea vehicles and drones. In recent years the use of unmanned surface vehicles (USVs) for military applications has also been the subject of investigation. One of the two test platforms used for this purpose is Odin, which is using water-jet propulsion and a wide range of sensors to navigate and analyze its surroundings. Odin is one of the resources used in mine countermeasures research at FFI due to its potential role in mine countermeasures operations and as support vessel for autonomous underwater vehicles.

The field of autonomy is a modern and developing field of research and exploration. Collision avoidance (COLAV) and motion planning are core topics in this research to be able to safely achieve fully autonomous vehicles. The problem is equivalent to the one helmsmen have been facing for centuries: *How do I steer this vessel in such a way that the situation at hand is resolved as smoothly as possible?* The problem is complex due to objectives which may be conflicting, such as path following and collision avoidance. If a vessel like Odin is to operate *autonomously* a wide range of requirements have to be met. One of these is to be able to interact with other vessels in regular seafaring situations. In the maritime domain, *The International Regulations for Preventing Collisions at Sea* (COLREGs) in addition to the constant quest for reduction of the risk of collision is governing how vessels should interact. Odin will be required to operate safely and efficiently in complex dynamic areas. It is advantageous for the vehicle to be able to plan its action ahead, increasing the likelihood of optimal vehicle trajectories. Thus, algorithms for motion planning for underactuated marine vehicles is highly relevant.

In this thesis, two low-level, sample based Model Predictive Control (MPC) COLAV algorithms have been investigated through simulations. Such algorithms function as a part of a layered architecture and have as their main purpose to handle emergency situations. In addition, it is desirable for them to comply with COLREGs. The first algorithm is the *branching-course model predictive control*-algorithm (BC-MPC) which was developed at *Department of Engineering Cybernetics* at the *Norwegian University of Science and Technology* (NTNU) by Bjørn Olav Eriksen. Its purpose is to always produce trajectories which causes the vessel to avoid collision and that are feasible with respect to the dynamic constraints of the vessel. The other is the *sample based model predictive control*-algorithm (SB-MPC) which is also developed at NTNU, by Tor Arne Johansen and Inger Berge Hagen. The algorithm produces course offsets and speed coefficient which can be combined with references from a higher-level planner thus producing resulting references. Both algorithms are able to prioritize when faced with diverging objectives such as path following and collision avoidance.

To evaluate the two algorithms, they have both been implemented on a 3 degree of freedom vessel model of Odin. A nominal course reference for navigation is provided by a Line-of-Sight (LOS) guidance system, while the nominal speed is user defined. The systems are asked to perform at equal situations and their performances are evaluated. The originators' tuning and implementation method is used as a basis for exploration and other possibilities for increasing performance are also explored and presented.

Several complex multi-obstacle scenarios where the obstacles fail to perform the necessary measures to comply with the COLREGs, thus causing dangerous situations are tested. The vessel must plan ahead, prioritize its conflicting objectives of path following and collision avoidance and create safe resulting maneuvers. In all scenarios the algorithms perform in such a way that collisions are avoided. Advantages and disadvantages from each of the algorithms are discussed.

Sammendrag

Forsvarets forskningsinstitutt (FFI) har i flere år forsket på undervannsfartøy og droner. De siste årene har fokuset også blitt rettet mot autonome overflatefartøy. En av testplattformene som brukes til dette formålet er *Odin.* Odin har sin fremdrift fra to vannjetter og er vel utstyrt med en sensorpark som analyserer sine omgivelser ved bruk av kamera, radar og lidar. Den er en del av ressursene FFI bruker til mine-mottiltaksforskningen grunnet dens potensielle rolle i mine-mottiltaksoperasjoner og som støttefartøy for autonome undervannsfartøy.

Autonomi er et moderne forskningsfelt med stor internasjonal interesse. Kollisjonsunngåelse og bevegelsesplanlegging er nøkkelemner i forskningen på vei mot trygge fullautonome kjøretøy. Innen det maritime er et av problemene likt det styrmenn i århundrer har møtt: *Hvordan styre på en tydelig måte som gjør at situasjonen blir løst hensiktsmessig?* Problemet er komplekst fordi det krever avveininger og vurderinger av ulike alternativer for å forsøke å oppnå flere, muligens motstridende målsetninger. Dette kan være banefølging og kollisjonsunngåelse. For at et fartøy som Odin skal kunne operere autonomt må systemet imøtekomme en rekke krav. Et av disse vil være evnen til å samhandle med andre skip i ordinære sjøfartssituasjoner, og dermed overholde sjøveisreglene, samtidig som kollisjonsfaren til enhver tid søkes redusert. Det kreves at Odin kan operere i komplekse dynamiske miljø på en trygg måte. Det vil være fordelaktig å kunne planlegge egne manøvre i forhold til andre fartøy for å øke sannsynligheten for optimal avvikling av situasjonene som oppstår. Dette gjør bevegelsesplanleggingsalgoritmer for underaktuerte overflatefartøyer aktuelle for utforskning.

I denne oppgaven blir to lavnivå *samplingsbaserte model predictive control*-algoritmer

(MPC) implementert på en skipsmodell av USVen Odin. Begge algoritmene er tiltenkt en rolle som en del av et fler-nivå kollisjonsunngåelses-system, og deres formål er å raskt kunne oppfatte og håndtere nødsituasjoner. I tillegg er det ønskelig om dette gjøres på en måte som er i tråd med sjøveisreglene. Den første algoritmen er *branching-course model predictive control*-algoritmen (BC-MPC) som er utviklet ved *Institutt for Teknisk Kybernetikk* ved *Norges Teknisk-Naturvitenskapelige Universitet* (NTNU) av Bjørn Olav Eriksen. Den har til hensikt å alltid produsere gjennomførbare referanser som reduserer risikoen for kollisjon og prioriterer fartøyets målsetninger. Den andre algoritmen er *sample based model predictive control*-algoritmen (SB-MPC) som også er utviklet ved NTNU, av Tor Arne Johansen og Inger Berge Hagen. Algoritmen produserer kurs-offsets og fartskoeffesienter som kan kombineres med referanser fra høy-nivå planleggere for å skape resulterende referanser. Begge algoritmene er kapable til å prioritere når de er møtt med divergerende målsetninger slik som banefølging og kollisjonsunngåelse.

For å evaluere algoritmenes egenskaper har begge blitt implementert på en modell av Odin med 3 frihetsgrader hvor skipets dynamikk simuleres. Nominelle kursreferanser produseres av et line-of-sight (LOS) guidance system, mens den nominelle hastigheten er gitt av brukeren. De to algoritmene sammenlignes ved at de blir testet i like situasjoner gjennom simuleringer. Den foreslåtte tuningen fra opphavsmennene har blitt brukt som utgangspunkt, før andre muligheter for å heve ytelsen ytterligere blir utforsket og presentert.

Flere komplekse scenarier med flere hindringer som ikke innretter seg etter sjøveisreglene og dermed skaper farlige situasjoner blir simulert. Algoritmene må planlegge fremover, veie målsetningene opp mot hverandre og skape referanser som er trygge å følge. I alle situasjonene er ytelsen av slik karakter at kollisjon blir unngått. Fordeler og ulemper med de ulike algoritmene blir presentert og diskutert. Avslutningsvis drøftes ulike tuning-parametres påvirkning på algoritmenes evne til å produsere trygge baner båten kan følge. Simuleringene validerer algoritmenes funksjon i de ulike scenariene og resultatene er tilfredsstillende.

Contents

List of Tables

List of Figures

Preface

This master's thesis is the culmination of the 2-year study program of *Industrial Cybernetics* at the *Norwegian University of Science and Technology* (NTNU). The thesis is the outcome of *TTK4900 - Engineering Cybernetics, Master's Thesis* and is valued with 30 credits. This project has been evolving around the topic of sample based model predictive control (MPC) for use in collision avoidance of unmanned surface vehicles (USVs). It has been conducted in Trondheim as collaboration between the *Norwegian Defense Research Establishment* (FFI) and NTNU.

At the initiation of the project I was provided with a 3 degree of freedom (DOF) Simulink model of the USV Odin by FFI. This contained the low-level controllers for heading and surge velocity, a waterjet model transforming control inputs to forces and moments and a hydrodynamic model of the vessel.

I would like to express my gratitude towards some of the people that have helped me in the process of producing this thesis. Prof. Kristin Y. Pettersen at the Department of Engineering Cybernetics, NTNU has supervised the project and provided great support and key insights. Dr. Martin Syre Wiig and M.Sc. Else-Line Ruud have been co-supervisors and have been great resources within recent research, implementation techniques and knowledge about Odin. I am grateful for the advice and feedback which have proved very valuable throughout the last year. Without it, the quality of the thesis would not have been as high.

The thesis addresses the family of sample based MPC algorithms and examines Dr. Bjørn Olav Eriksen's *branching-course MPC* (SB-MPC) and M.Sc. Inger Berge Hagen's *sample based MPC* (SB-MPC). I am grateful for the input and support

received from both originators as they have been welcoming and provided me with first hand experience and insights from their work with the algorithms.

Unless otherwise stated, all figures and illustrations have been created by the author. This is exciting work which, I hope, in the long run will be able to contribute towards FFI's goal in developing the future autonomous maritime mine counter measure (MMCM) capabilities.

Torstein Fagerlund

Trondheim, June 2020

Chapter 1

Introduction

In this introductory chapter an overview of the background and motivation for the thesis is presented. A list of assumptions which are applied to limit the scope is given. The contributions of the thesis are put in the context of previous work from the specialization project [1]. Finally, a presentation of the structure of the thesis is given.

1.1 Motivation

The automation of tasks in the society has been an objective across sectors for decades. In many applications the purpose of automation is to increase the safety and reduce costs. Automation is ranging from simple functions such as self-opening doors, to more complex self-governing, autonomous systems such as self-driving cars. Such technological progress enables people to avoid many boring or dangerous tasks, allowing these to be performed with little human involvement. Boring tasks are typically tasks that are repetitive or non-challenging and trivial. While a human operator may lose interest and focus, thus inflicting his performance, a machine would be able to maintain its performance over time, thus increasing the safety of a system.

The *Norwegian Defence Research Establishment* is responsible for defence related research in Norway and has been researching autonomous robotics for several

Figure 1.1: Odin underway. Courtesy of FFI.

years. One of the fields of investigation is the operation of unmanned surface vehicles (USVs). The ability to employ USVs lay the foundation towards the long-term goal of a new generation of equipment and methods for *mine counter measures (MCM)* [2]. By 2028, the goal is to have a new system in place for autonomous detection and defusage of sea mines. The purpose is to move operators out of the danger of mine fields. In this new era; human intervention in a potential minefield should be superfluous such that in the event of an accident human lives will not be at direct stake. In the recent years Odin has been used as a research platform for autonomy in the Norwegian Defence' capabilities.

The *Royal Norwegian Navy (RNoN)* already uses the autonomous unmanned vehicle (AUV) HUGIN for detection, localization and classification of sea mines [3]. A remotely operated single use weapon is used for the destruction of mines. A mothership at a safe distance can be used to deploy small and fast USVs that carry the AUVs and weapons into the operation area. The USV also acts as a relay for communication between the mothership and AUV during the operation. Sea mines are still legal according to international law, and their use is widespread around the world [4]. As a part of NATO and as a nation with a long coastline it is important for Norway's sovereignty to be able to maintain control over these areas and to be in the forefront of maritime technology. This enables us to maintain our position as a nation with great nautical history and tradition. One of the

actions needed to achieve this goal is to develop our own capabilities in maritime defence technologies. It is now the job of both the Norwegian Defence Research Establishment (FFI) and the RNoN to further develop concepts and solutions for autonomous *maritime mine counter measures (MMCM)* that are deployable from a safe location outside the minefield.

In addition to the clear need for autonomy in the field of MMCM, collision avoidance and autonomy at sea are very current topics. FFI's USVs need to be able to handle regular seafaring situations they encounter as well. In the maritime domain of autonomous transportation several uncertainties such as obstacles with uncertain position and speed are influencing the situations. Other vessels' course of action can also cause uncertainty in the decision making. External forces such as the current sea state and weather may also affect the situations greatly and demand a robust control system.

The development of autonomous collision avoidance has started with hybrid systems, meaning systems that aid the helmsman with situational awareness and suggestions of alternative actions. Recent development in both sensor systems and computational power has allowed for more research and development in the field. The objective of this thesis is to investigate the use of collision avoidance algorithms for use on the vessel model of Odin. These algorithms are able to plan ahead to create more optimal behavior and increased safety.

1.2 Assumptions

The following general assumptions have been made in the work with this project:

1. The system is considered fully observable. All values which in reality may have to be estimated, may be used in calculations.
2. The position, velocity and orientation of the obstacles are known.
3. Obstacles are assumed to move in a straight-line path with constant speed.
4. The situations take place in open sea, meaning that the vessel may assume sufficient depth under its hull at all times and locations.
5. All situations occur on the vessel's straight-line path towards the next waypoint which is considered to be far away.
6. The vessel is assumed to have a reasonably functioning low-level control

system which causes provided references to be followed.
7. The calculation and selection of optimal trajectories are made in a single step of calculation time.
8. We assume that the vehicle will remain in the displacement phase, such that the simulations with the vessel model may represent real performance in a good way.
9. No wind, waves, current or other external forces are affecting the vessels.

Remarks:
The collision avoidance situations occur far from the next waypoint (WP), thus avoiding the need for heuristics that would make the vessel cope with *not* passing the WP. Maybe the vessel is well off by passing the WP and heading for the next one, or maybe the mission is dependent on the vessel passing the WP. This decision is up to a higher level planner to decide. In real life the boat would have travelled some distance in the duration of the calculation time. The challenges this presents are considered to be beyond the scope of this project and implementation techniques would need to be applied. The vessel is assumed to be able to follow all reasonable provided references because of the assumption of the well functioning and verified control system. This assumption allows for the prediction of future states, given a certain control input. However, the investigation of the low-level control system is outside the scope of this thesis.

In the *branching-course model predictive control* (BC-MPC) algorithm, the nominal line of sight trajectory is assuming an instantaneous turn and acceleration as it is produced without thought to vessel pose and dynamics. In the *sample based model predictive control* (SB-MPC) algorithm, however, all the trajectories are calculated without taking these aspects into account, thus assuming an instantaneous turn and acceleration.

1.3 Previous work

In the fall semester of 2019, a 7.5 credit specialization project, *TTK4551 - Engineering Cybernetics, Specialization Project*, was completed under the supervision of Kristin Y. Pettersen at NTNU and Martin S. Wiig and Else-Line Ruud at FFI. The resulting thesis was named *Motion planning and collision avoidance for USVs* and evolved around Odin and the BC-MPC algorithm. The main results of the specialization

project [1] consist of the following elements:

- The framework for being able to test different collision avoidance situations
- The implementation of the constant speed BC-MPC algorithm in MATLAB.
- Systematic simulation and evaluation of the effect of BC-MPC tree generation parameters.
- Implementation and evaluation of two different cost functions.
- Simulation and evaluation of the performance of the constant velocity BC-MPC algorithm on the Odin model.

1.4 Contributions

The main contributions of the work presented in this thesis are as follows:

- An extensive literature review on motion planning and collision avoidance.
- The implementation of both the BC-MPC and the SB-MPC algorithm on the Odin vessel model in MATLAB and Simulink.
- A method for increased accuracy in the trajectory tracking for the BC-MPC algorithm for feedback low-level controllers.
- The proposal of a new function for the evaluation of *cost of collision* in the SB-MPC cost function.
- The proposal of a new align function and linear decrease of penalty in the avoid function for the cost function with associated tuning in the BC-MPC algorithm.
- A simulation study evaluating the performance of the two collision avoidance algorithms in regards to the COLREGs and a safety perspective.

1.5 Outline

The remainder of this paper is structured as follows. First, in Chapter 2 an extensive literature review on *motion planning*, collision avoidance, key parts of *The International Regulations for Preventing Collisions at Sea* (COLREGs) and associated algorithms is presented. Then, in Chapter 3 presents the system description of Odin, its controllers and the sideslip effect. The *line of sight* LOS guidance scheme for path following is also presented along with the method for predication of obstacle positioning. The trajectory generation in the BC-MPC algorithm is presented in

the first parts of Chapter 4. The chapter also presents an extensive cost function for optimization and trajectory selection. Chapter 5 is devoted to the presentation of the SB-MPC algorithm and is including both the tree generation and selection. Next, in Chapter 6 different practical adjustments of the algorithms and their tuning are presented. The tree generation and cost functions of both algorithms are discussed. Then, the two algorithms are used in the simulations with the vessel model of Odin in five single obstacle, and three complex multi obstacle scenarios in Chapter 7. Finishing, the performance of the two algorithms are discussed in the light of the simulations. Concluding remarks, important lessons learned and suggestions for further work is presented in Chapter 8.

Chapter 2

Literature review

In this chapter relevant topics and past achievements in the field of collision avoidance and motion planning will be addressed. The parts regarding collision avoidance and COLREGs has its roots from the specialization project.

2.1 Collision Avoidance

Collision avoidance (COLAV) in the maritime domain has historically been handled by the crew operating the ship. It must sense and interpret the situation at hand, process it, and come up with a plan to solve the situation. *The International Regulations for Preventing Collisions at Sea* (COLREGs) establish how one should act in different situations. The situation is a multi-variable nonlinear problem that traditionally has been solved by reviewing the position, speed and course of the involved ships while taking a map of the immediate environment, as well as the current weather, into consideration. The helmsman may alter both his course and velocity in order to resolve a potential situation. In more recent times helmsmen are aided by sophisticated technical solutions. One is the option to communicate with the other party directly with a *very high frequency* (VHF) radio. This is a valuable addition to the traditional, indirect communication through actions and maneuvers.

Another semi-modern system is the *automatic identification system* (AIS). The AIS

system broadcasts position and other relevant data over the VHF-network [5]. This information is by some boats and some transition-stations published on the internet to web-pages like "marinetraffic.com". The frequency of updates over the AIS-system is varying and can range up to once per minute. The distance from shore and other vessels also inflicts how well such a signal may be sent and received. In Figure 2.1 a view at how information from the AIS system may be displayed is presented. One can see the four ferries operating the stretch between Horten and Moss. When the ships are going east, they take a more southerly route than when they are heading west. This way they are always keeping right of each other and operate in a predictive and easy way. Unfortunately, one cannot assume that all vessels are using the AIS and VHF systems, therefore an autonomous collision avoidance system cannot fully depend on these systems.

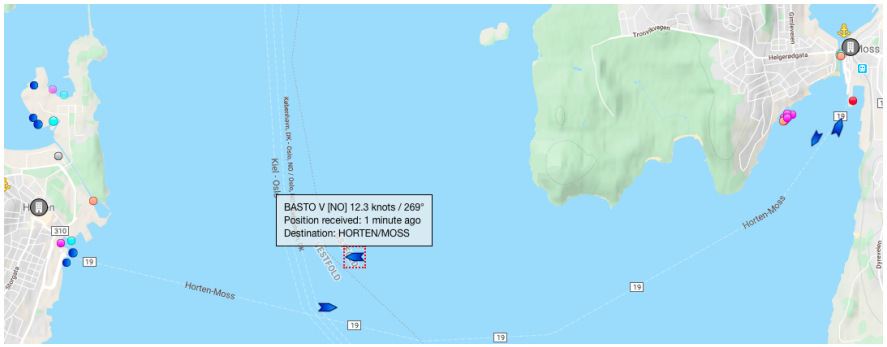


Figure 2.1: AIS information from the coastal area between Horten and Moss in Norway displayed on a map. Screenshot from marinetraffic.com [6]

2.2 Collision avoidance regulations - COLREGs

There are many different measurements that could give an indication of how serious a situation is, with regards to a potential collision. These may be; distance between ships, time to collision, Closest Point of Approach (CPA), Time to Closest Point of Approach (TCPA). What kind of limits that would be reasonable to say that the ship should start to react to may vary. If two ships encounter each other in open waters, it may be reasonable to make sure that one steers clear of the other by a larger margin than one could hope to achieve in more restricted waters. Which velocities they are travelling in and which collision avoidance rules that come into play are also parameters that affect how the situation should be interpreted and handled.

Two helmsmen may also react differently to the same type of situation.

The International Regulations for Preventing Collisions at Sea (COLREGs) are published by the *International Maritime Organization* (IMO). These rules originate from far back in time, but the current set of rules are from 1972. These rules describe how different situations at sea should be solved, how one should steer a vessel and how responsibilities are partitioned. Complex multi vessel situations could occur, where one must interpret which course of action that seems to be the most reasonable. The rules are built such that one sometimes must *break* a rule in order to *obey* another.

Further, one can never achieve a right over another vessel. Vessels are rather categorized as either a "give way" (burdened) vessel and a "stand on" (privileged) vessel. A situation may have two *give way*-vessels with no *stand on* vessel. In a situation with three vessels one could be the *give way* to two of the others, while one is the *give way* to only one, and the last one is simply a *stand on*-vessel to both.

The COLREGs are built up by Parts A to F where each part contains rules that are in similar categories. Part B - which is concerning *steering and sailing*, is again partitioned into 3 sections. In addition, there are also 5 appendices. Many of these rules are general and regarding equipment, communication and seamanship. In regard to *collision avoidance systems*, Part B Section II is the most relevant. It governs rules of steering and sailing while two ships have each other in sight. Rules 11 and 12 is governing sailing vessels, while rules 13-17 are more general. Many of the rules are describing how responsibility is divided among the vessels as a function of their position in regard to each other. Figure 2.2 displays a way the vessels may be classified.

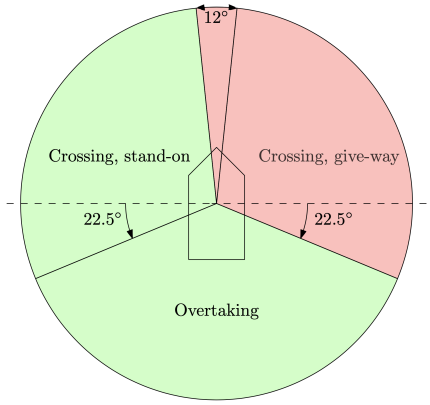


Figure 2.2: A top view of own vessel which classifies the sectors another boat may be classified by. [7]

In this thesis the ability to comply with COLREGs is the subject of investigation, but in a long-term view, one may look at COLREGs to be able to see the standard of complexity in maritime traffic.

2.2.1 General remarks

These rules describe 1:1 vessel situations and are open for interpretation. *Rule 6 - Safe speed*, states that it is a vessel's obligation always to maintain a *safe speed*, to avoid collision and to be able to steer clear of or stop in front of obstacles. The speed chosen shall be a result of the nature of the environment, surrounding waters, the visibility, the maneuverability of the vessel, weather and traffic density. *Rule 7* regarding *risk of collision* impose all vessels to continuously evaluate the presence of risk of collision. It states that factors that determine how large such a risk is viz. a constant compass bearing of an approaching vessel, and when the distance to the other vessel becomes substantial in relation to the size of the other vessel. *Rule 8 - Action to avoid a collision* demands that 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.

2.2.2 Rule 13 - Overtaking

Rule 13 states that any vessel overtaking any other shall keep out of the way of the vessel being overtaken. A vessel is defined as overtaking if it is coming from

behind from a direction more than 22.5° abaft her beam. The overtaking vessel may choose itself how it is to keep out of the overtaken vessel's way. Once the situation is defined as an overtaking situation, it shall not be interpreted as a crossing situation later on, even if the positions of the vessels may allow one to think that is so.

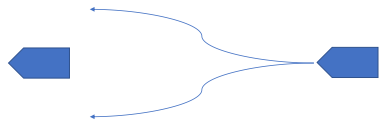


Figure 2.3: Overtaking

2.2.3 Rule 14 - Head-on situation

Rule 14 states that when two vessels are facing each other head on, with reciprocal courses and they both are burdened vessels which share the responsibility of keeping clear, they shall both alter their courses to starboard and pass on the port side of each other.

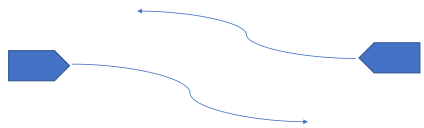


Figure 2.4: Head-on

2.2.4 Rule 15 - Crossing

Rule 15 is governing the crossing situation and declares that the vessel which has the other to her starboard shall keep out of the way, and if the circumstances allow it, she shall pass aft of the other vessel. This therefore implies that the vessel which has the other boat to her port is the stand-on vessel and is not burdened by the situation.

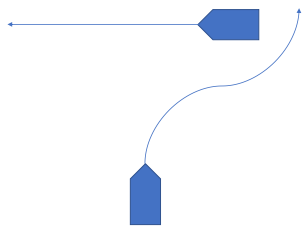


Figure 2.5: Crossing

2.2.5 Rule 16 - Action by give-way vessel

A vessel that is directed to give way shall perform its action to keep clear early and clearly, so that the other vessel may observe it.

2.2.6 Rule 17 - Action by stand-on vessel

The stand-on vessel is obliged to maintain its course and speed so that the situation at hand may be resolved in a predictable way. However, if the give-way vessel fails to take appropriate action and a collision cannot be avoided by the action of the give-way vessel alone, then the stand-on vessel shall take action to best resolve the situation and avoid collision.

2.3 Motion planning

The field of *motion planning* evolves around the problem of finding the best possible way to configure an object or robot through space over the duration of a time-window. Thus, the object is made to move from its origin pose (i.e. position and orientation) to a goal pose. This is also known as the *piano mover's problem.* A

motion planning algorithm may include knowledge about the robot's abilities to move, i.e. constraints, the stationary environment (a map) and sensor data about moving obstacles.

How FFI divides the works with autonomy in its divisions sheds some light on the associated topics. Scene analysis, swarm, and automated missions are the three divisions. Control and planning are topics which are closely bounded and are sorted into the automated missions division. Scene analysis is the field of perceiving the vessel's surroundings through sensors. It is a common simplification in route planning to simply assume zero error in the scene analysis.

2.3.1 Algorithms

Canny and Reif showed in 1987 that the problem of dynamic motion planning for a point in a plane, with obstacles and bounds on states such as velocity, is NP-hard [8]. In practice it turns out to be a lot of different heuristics, which simplifies the problem, and which often results in good practical performance.

One way of categorizing the algorithms in the field of motion planning and collision avoidance is by separating between *reactive* and *deliberate* algorithms. Reactive are of the *sense-act* kind, while deliberate algorithms are of the *sense-plan-act* kind.

The *artificial potential field* was first presented by Khatib in 1986 [9], and models the obstacles as a repulsive field and the goal as an attractive field. These fields are summed to provide a resultant virtual force acting upon the vehicle, thus moving it around obstacles and towards the vehicle's goal. This method is closely related to the *virtual force field method* [10], and is intuitive, easy to implement and has a low computational complexity. However, the method creates a nonlinear field which is prone to local minima, and the lack of planning ahead can lead to suboptimal and oscillatory vehicle trajectories.

The *Velocity Obstacle* (VO) is a *reactive* algorithm invented by Fiorini and Schiller [11] which is a very common and acknowledged method for collision avoidance in dynamic environments. All velocities of a robot that cause a collision with another robot form the set of a velocity obstacle, assuming that the other robot maintains constant velocity. Having calculated which velocities that would cause a

collision, one also subtracts infeasible velocities by using the vessel/robot dynamics resulting in a set of reachable avoidance velocities to choose from. From these the VO-approach does not dictate how a velocity is selected, but the use of a cost function could decide this. The VO-approach avoids local minima, but may cause sub-optimal behavior in complex scenarios. This is due to the algorithms lack of planning and ability to look forward in time. Having several competing goal or objectives then may cause problems. Planning algorithms therefore become a natural next step of exploration.

Model Predictive Control (MPC) is an attractive choice of control principle which may handle such complex problems [12]. MPC is a way of closing the loop of an open loop optimization problem. The model predictive controller evaluates the current states and select the optimal input to control the system in the desired direction. The system responds to the input and the new states are then used to solve the problem again yielding another set of new states. A quite interesting point regarding the MPC regime is that the optimal inputs and states are calculated for the whole prediction horizon, but only the first control input is actually applied. It is possible to have algorithms that take longer time, which feed the system with a short sequence of control inputs, before another solution is calculated and ready. MPC have traditionally been applied in process regulation with fairly slow dynamics and long prediction horizons [13]. The computational complexity correlates with the run-time it takes to solve it. If you compute a very accurate model it takes longer time to compute, and therefore longer time before the model can receive a new - updated input. There is a trade-off between being able to calculate newer solutions faster and being able to see far into the future in an accurate way, to get optimal inputs. Figure 2.6 shows an illustration of the MPC principle.

Nonlinear MPC is simple in concept, but the run-time associated with finding the optimal solution may cause problems. A *nonlinear program* (NLP)-solver is needed in such a case, and both *sequential quadratic programming* (SQP) and *interior point* methods may be applied, but it is very difficult to guarantee for the optimality of such a solution. Due to the nature of the problem of motion planning being a highly complex problem, *sample based MPC* approaches become attractive. Sample based MPC approaches have proven to be able to both avoid local minima and avoid causing computations to last too long due to the finite number of alternatives

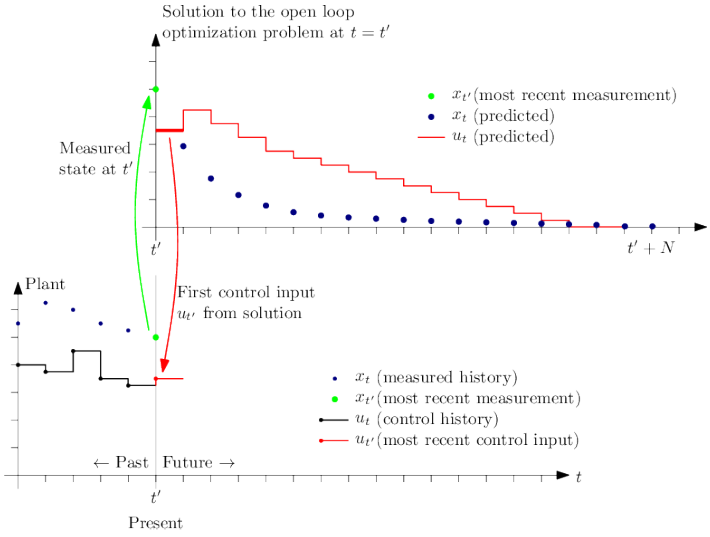


Figure 2.6: MPC principle illustrated. Figure from [13]

which are evaluated. Instead of calculating the optimal input in a continuous search space, sample based MPC rather calculates a value indicating how well each candidate in a set of candidates performs. The one that scores the best function value from the cost function is then selected and passed on to the system. What is lost in the transition from solving the problem in the continuous search space to in a sample based search space is the possibility of proving optimality. However, in the field of maritime navigation and control, the options a helmsman would be having, may be well represented by making the discretization of the search space fine-meshed. Two implementations of such sample based algorithms are; Eriksen's BC-MPC and Hagen's SB-MPC.

Although *sample based MPC* may be applied for a group of algorithms, the name is in this thesis linked to the specific algorithm presented in [14], [15] and [16], and referred to as the SB-MPC algorithm.

With today's available computational power, planning algorithms have become more attractive, as the problems may increase both in complexity and in planning horizon. This enables the users to achieve more optimal solutions in their pursuit to achieve reliable anti-collision systems as a part of their autonomous solution. Recent development in computational power allows solving larger problems faster,

and thereby calling them even more often. The planning involves calculations into different alternatives of actions and evaluation of these. Previously, deliberate algorithms where only explored academically and where exclusively used for pre-operation purposes with a priori information. But now, the group of algorithms is considered applicable for online collision avoidance even though the environment is characterized by quick, unpredictable and frequent changes in the environment. This information comes from *sensory*-information. Due to the increased computational power, the separation between the two categories is therefore no longer as substantial, and deliberate algorithms may prove valuable for online collision avoidance.

Another way to distinguish between motion planning algorithms is by global and local algorithms [17]. Global algorithms seek to solve the entire problem from start to goal, while local algorithms seek to provide a part of the solution by reviewing a local, defined part of the total space. Typically, global algorithms are applied before a mission, and has little constrain on the computational complexity and therefore computation time. On the other hand, local algorithms may take choices online based on the current situation around the robot the algorithm controls, thus putting large demands on the computational complexity of the problem. It is possible that the local algorithms perform choices that are, when seen globally, not optimal, however, the algorithm may have chosen it as a result of a greedy policy. A greedy policy is a mapping between a situation or a state and a resulting action. A greedy policy is a policy that chooses its action based on what is currently available of information, which for a local algorithm may be limited.

One algorithm which is planning from start to finish is the *Probabilistic roadmap* (PRM). It works by generating a random line segment and checking if the path of the segment is either clear or contains a collision with an obstacle. If a collision is present then the segment is rejected [18]. In the exploration of a cluttered environment the algorithm spends most of its time doing collision detection.

The *rapidly-exploring random tree* (RRT) algorithm is an algorithm that determines the control inputs that can cause a robot to move from an initial- to a goal-configuration while obeying the dynamic model and avoiding collision with obstacles in the environment. Lavalle and Kaufner [19] further developed the already known *random tree*, into the RRT. Both algorithms generate a random node

in space and a step is taken from a point in the tree in the direction of the new node. The random tree does so from a random node in the current tree, but the RRT generates an edge from the nearest node in the current tree in the direction of the new node. This causes the trees to develop drastically differently since RRT is now growing into the unexplored at a much higher rate. The path from start to goal may be discovered more quickly. However, observing such a path, one can see that it is probably not the shortest path from start to finish. And if one where to continue adding new nodes to the tree, it would still not result in a shorter or more optimal path, since the edges between the nodes are static.

In the formulation of such problems one can easily associate a cost of travelling over an edge from one node to another, with the distance between these nodes. Other parameters which may affect the physical situation of doing so may also be incorporated into such a cost function. Karaman and Frazzoli invented the RRT* which are provably asymptotically optimal, meaning that the cost of the solution converges to the optimum as the number of iterations performed by the algorithm increases [20] [21]. It works by re-evaluating the cost of reaching each node in the tree at given intervals, using a shortest path-search. If a shorter path to a node is found, the edges are rewired to this new path. The advantage of optimal paths is paid for in more "overhead" calculations, causing the whole algorithm to be slower.

Sampling based algorithms are unable to determine that there doesn't exists a feasible path. PRM and RRT have been shown to work well in practical applications and also carries probabilistic completeness. This means that the methods' probability of finding a solution approaches 100% as the computation time becomes sufficiently large, however, the methods are incapable of determining whether or not a solution exists. The PRM, Random tree, RRT and RRT* are all probabilistic complete.

RRT and RRT* are examples of general, global algorithms employing randomness in their exploration. Other algorithms, such as the BC-MPC and SB-MPC algorithms, are *local* algorithms which use a predefined, horizon-limited tree of feasible path alternatives. Each of the candidates in the tree are then are explored and tested for their predicted performance.

Brooks et. al. [22] propose a method for tree generation in local parts of the

environment by exploiting randomized sampling for exploration in a way not too unlike RRT. The algorithm focuses on the motion planning problem using only local information, and optimizing the path, with a cost function designed to bring the robot closer to the long-term goal, over a limited time horizon. This is applicable and useful as large parts of the environment will be outside the sensor range of many moving robots. The branching of the tree is based on the end position from the best control trajectory from the previous iteration. A finding from this work which is interesting is the pruning of the tree if a node is evaluated to be of a certain poor quality, e.g. a collision. This allows for large parts of the state-space to be ignored in further computation, resulting in faster convergence towards an optimal path. The method has proved functioning for both a car and a Segway operating in walking pace.

Grid based algorithms aim to solve the problem of motion planning by discretizing the continuous configuration space into cells in a grid. Each cell is associated with a possible configuration and the neighboring cells may be seen as feasible "next steps" in a candidate path. Each cell may be assigned a value corresponding to how well the configuration is. Search algorithms like A* may be inputted these situations and grids and produce a path from start to goal configuration. The quality of the produced paths may be dependent on the granulation of the grid and the ability of the robot to perform the planned path. For high performance the grid size must be low, resulting in many cells with high computational complexity. One challenge may be to use the robot-dynamics-model to calculate which neighboring cells are possible to travel to, and still maintain the "stamp" of optimality from the mathematical deduction of the path.

2.4 Layered software architecture for motion planning

The purpose of having a layered motion planning software architecture is to be able to organize the flow of information and control inputs in a system with the purpose of dividing the computation of tasks and decisions into adequate portions. There are several reasons and advantages for choosing this type of software architecture. One of them is the ability to develop and manage each layer individually. This avoids building one single, large and complex block. In the field of computer science,

the approach of layered software has been an advancing preferred approach in many applications in the recent years. These applications range from secure payment solutions to web-page designs and to embedded control systems with online computation constraints.

Within the world of collision avoidance and motion planning, a constant challenge is the changing situation and environment of the robot. However, as the global computation of a path from start to finish is a multi-variable, nonlinear complex problem which requires a fair portion of computation, and therefore time, one faces the problem of an outdated solution, when it is ready for appliance. Tiered segregation allows for each layer to solve different parts of the greater overall problem and allows a short term COLAV algorithm to constantly "check" for obstacles and evaluate different control inputs to deal with them. A slower higher-level planner may evaluate a more complex problem over a larger horizon and a larger surrounding area for example also taking a sea-map, weather and current into consideration. This could be optimizing the nominal path for fuel efficiency and minimization of travel distance.

A mid-level collision avoidance layer could also employ the principle of MPC. However, this layer must solve a larger nonlinear problem in continuous time. As this is prone to local minima it is difficult both to guarantee optimality and to guarantee a worst-case computation time [23]. The algorithm is expected to be able to plan long-term maneuvers and at the same time be able to handle rapid changes to the environment. The low-level planner such as BC-MPC and SB-MPC or a reactive layer such as a VO-algorithm can be expected to be able to keep the vessel out of collisions as they are faster algorithms with such as their sole purpose.

In Figure 2.7 one can see a vessel faced with a map with some moving obstacles and a land area. Two alternate nominal paths are suggested and as this is a large problem with a long horizon it may be the job of a higher-level planner to decide on such a trajectory. It is not necessarily so that the computation of a such a problem is performed with the information regarding the moving obstacles at all. This would probably cause our vessel to choose route A. In reality a trained skipper or an algorithm considering those obstacles would maybe choose to travel the slightly longer but less busy seaway, alternative B.

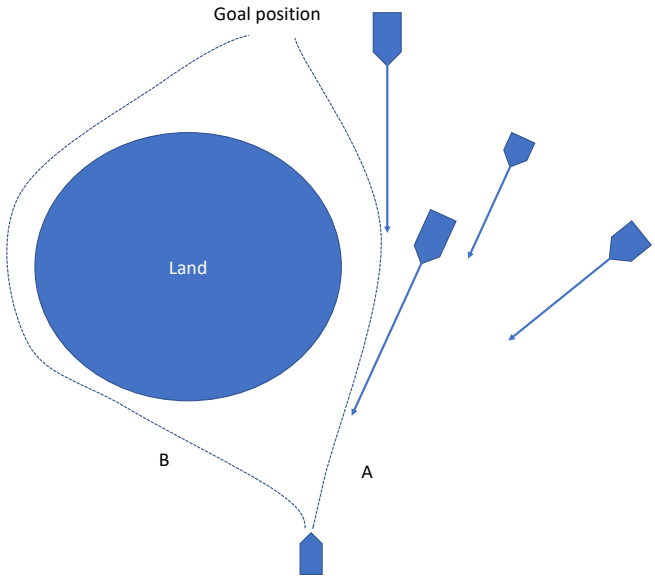


Figure 2.7: Motion-planning situation

2.5 Initial comparison of SB-MPC and BC-MPC

Both BC-MPC [24] and SB-MPC [15], are algorithms in the family of sample based MPC algorithms for collision avoidance. They are employed as a layer in a layered architecture for motion planning and collision avoidance systems. The purpose is to distribute the responsibility of handling different objects and obstacles so that the vessel may behave in a way that enables it to reach its long-term goal as well as reacting quickly to more short-term situations. A high-level planner will typically take care of static obstacles and navigation resulting in a nominal path. The planner may run prior to the vessel beginning its mission. Both algorithms are examples of low-level planners which are deliberate algorithms of the *sense-plan-act*-kind. The important task of this part of the hierarchy is the handling of current and rapidly changing dangerous events. BC-MPC and SB-MPC both makes a greedy, local choice which is likely to bring the robot closer to its goal. The difference in the two lie primarily in the assumptions that are made to be able to generate predicted trajectories and in the complexity, and sophistication of the trajectory candidates. BC-MPC is in general more complex than SB-MPC. Each algorithm is

equipped with its own cost function which holds several similarities compared to each other.

One advantage of the SB-MPC is the ability to be compatible with any other higher-level planner which provides a speed and course reference. This is due to the SB-MPC's way of scaling the speed and adding an offset to the course to be able to avoid collisions. Unfortunately, if the higher-level planner is finished with one line-segment and then changes its course, the course reference will be falsely generated and will be made out of a combination of two values generated independently from each other. If the time from one call of SB-MPC to another is short, the problem will be small and resolved fairly quickly. Hence, the time between calls to the SB MPC algorithm should be short, in the order of 5-10 seconds depending on the operation and speed of the vehicle. In this thesis the investigation of this particular problem is seen to be outside of the scope and is not investigated further.

One advantage of the BC-MPC algorithm is the lack of logic statements in the cost function. It is not desirable for a low-level COLAV algorithm to have distinct thresholds decide whether to go for plan A or B. The strive to avoid discontinuities and logic statements intend to improve the robustness in respect to obstacle noise. The cost function components of BC-MPC awards choices that are similar to the previous choice. Another aspect is for the horizon in low-level algorithms not to become too long, due to the lack of COLREGs-specific logic in the algorithm.

For the purpose of comparing these two algorithms they will in Chapter 7 be used to resolve the same situations, and advantages and weaknesses will be discussed. Many of the parameters in the two algorithms will be set equal to each other so that it is not the effect of for example difference in horizon-length that causes these differences. SB-MPC may be added on to a maybe unknown hybrid architecture, while BC-MPC is in a larger degree built into the higher-level planners.

Chapter 3

System description

In this chapter a description of the Odin USV, the vehicle model and the system architecture is provided. Several topics in this masters thesis is coinciding with the topics from the specialization project [1]. Equations and descriptions of the vessel, its controllers and effects such as sideslip are therefore in some way reused and is to a large degree inspired by the specialization thesis.

3.1 The Unmanned Surface Vessel - Odin

The Odin USV is an unmanned surface vessel (USV) FFI uses for research and development purposes. Odin was the first vessel and it is now accompanied by Frigg, which is very similar. The vehicles may be controlled manually, but the main purpose is to develop and discover systems enabling autonomy on board.

The software onboard employs the *Robotic Operating System* (ROS) which acts as a middleware between sensor drivers, the operator interface and the control system components. Odin is a rigid buoyant boat (RBB) with propulsion from two parallel mounted waterjets. Figure 3.1 shows the stern of Odin and its two deflectors affecting the continuous stream of water. Simplified, they draw water from underneath the boat and push it out the back. To steer the boat, one can change the angle of the waterjet nozzle. To make the vehicle move backwards, a deflector is used to steer the water flow from the nozzle forwards, thereby pushing the boat

Figure 3.1: Odins two waterjets and deflectors seen at the stern of the vessel. Courtesy of FFI.

backwards. The double waterjet system enables 3 DOF control as the controller can inflict a wide variety of forces and moments on to the hull which allows for very good maneuverability. This is essential for accurate position keeping or docking operations, but when the boat is used in more traditional operations, like transit at maneuvering speeds, the deflector is typically raised all the way and one controls the boat with a combination of the magnitude of the force and the direction it is pushed resulting in 2 DOF control in surge and yaw. This allows for the boat to function as if it is propeller driven, and it performs with the limitation of being an *under-actuated* system.

The dynamic model of Odin developed by FFI is a 3 DOF model containing the states surge, sway and yaw. In layman terms - forward velocity, sideways velocity and heading direction. Such horizontal-plane models are commonly used for ships where the path following or trajectory tracking capabilities are investigated [25]. Figure 3.2 shows the motion terminology in 3 degrees of freedom.

The kinetic equations of motion are described in the following way [25]:

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{3.1}$$

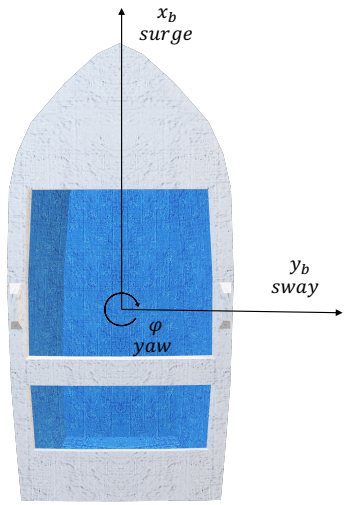


Figure 3.2: Top-view of a vessel marked with motion-terminology in 3 degrees of freedom (DOF)

$$\boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{3.2}$$

Where u,v and r is the surge, sway and yaw-rate, respectively.

M is the mass matrix, C is the coriolis matrix, D is the damping matrix and τ the forces inflicted by the propulsion system [2] [25].

$$\mathbf{M} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix} \tag{3.3a}$$

$$\mathbf{C}(\nu) = \begin{bmatrix} 0 & 0 & c_{13}(\nu) \\ 0 & 0 & c_{23}(\nu) \\ c_{31} & c_{32}(\nu) & 0 \end{bmatrix} \tag{3.3b}$$

$$\mathbf{D}(\nu) = \begin{bmatrix} d_{11}(\nu) & 0 & 0 \\ 0 & d_{22}(\nu) & 0 \\ 0 & 0 & d_{33}(\nu) \end{bmatrix} \tag{3.3c}$$

$$\boldsymbol{\tau} = f(\delta_{\text{steering}}, \delta_{\text{throttle}}, \delta_{\text{deflector}}, \boldsymbol{\nu}) = \begin{bmatrix} \tau_u \\ 0 \\ \tau_r \end{bmatrix} \tag{3.3d}$$

Where τ_u and τ_r are the actuator forces and moments which act to cause the surge and yawing motions. As the second element in $\boldsymbol{\tau}$ is 0, the vessel is underactuated. meaning that the number of actuators is lower than the states. An underactuated system is a system whose states may not all be controlled by its actuators. In the case of many boats; surge and yaw may be controlled through the thrust and rudder, while the sway motion is induced by the surge and yaw motions. Sway is simply a uncontrollable state, which is usually assumed to be stable. The function defining $\boldsymbol{\tau}$, which is the waterjet model is developed by FFI and is not disclosed.

Note that this model does not contain any disturbances from e.g. ocean currents or waves. The model parameters for the simulation of the vessel dynamics were

provided by FFI. The vessel model is fairly accurately describing the real boat's behavior from approximately 2 to 6 m/s. Outside these boundaries other models for resisting forces and damping from water and air resistance on to the hull may be applied, as the vehicle then enters the planing regime. As it is not within the scope of this project to investigate the model, we assume that the vehicle will remain in the displacement phase, and have ensured that the speed is approximately 5 m/s in the simulations in Section 7.

The 3 DOF vessel model is described as follows:

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{3.4}$$

Where:

$$\boldsymbol{\eta} = \begin{bmatrix} N \\ E \\ \psi \end{bmatrix} \qquad \boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ r \end{bmatrix} \qquad \mathbf{R}(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

3.2 System architecture

The system architecture is built up of several block which each has their own function. The user inputs a nominal speed and a set of waypoints. Such information could also come from a higher-level planner. A LOS guidance law is employed to create a nominal course reference which doesn't take obstacles into account. This information in addition to the system states which are fed back from the vessel model are used in the low-level collision avoidance algorithm. This collision avoidance block is discussed in great detail and is the main topic of investigation in the thesis. The low-level controllers convert a speed and course reference into surge and heading references which are fed into a PD and a PID controller for heading and surge, respectively. This is converted into steering, throttle and deflector angle commands which are presented to the vessel model of Odin. The whole system may be seen in Figure 3.3. The system is considered fully observable without any measurement noise or disturbances, and key values are fed back to several parts of the system.

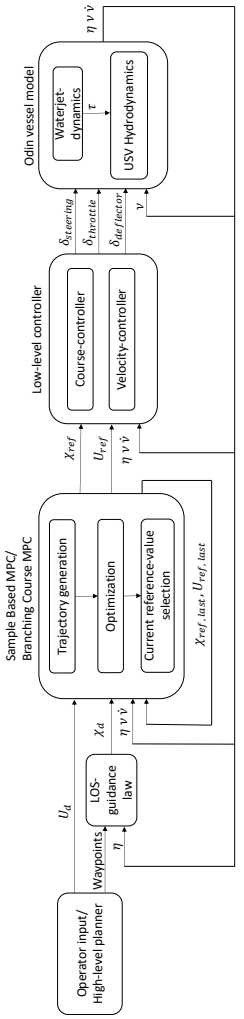


Figure 3.3: System architecture

3.3 Low-level controllers

The 3 DOF model of Odin with its controllers is originally designed to receive a heading and surge reference. Collision avoidance algorithms are often working around course and speed as this is more directly translated into positions.

The relation between commanded speed and surge reference is here presented. The speed U is commanded by either SB-MPC or BC-MPC and in the transition from a speed reference to a surge reference the following geometric relationship is employed; $U^2 = u^2 + v^2$, meaning that the speed is a combination of surge and sway motions for a 3 DOF model. Using this relationship with a commanded speed to create a reference for surge causes the following expression: $u_{\text{ref}} = \sqrt{U_{\text{ref}}^2 - v^2}$. The problem with this relationship is when U_{ref} is commanded to be lower than the current sway motion, for example zero. u_{ref} then becomes the square root of a negative value which would cause imaginary numbers. To ensure that the reference signal is well defined, u_{ref} is set to U_{ref} when $U_{\text{ref}} < u_{\text{ref}}$. This way, U is pushed towards U_{ref} as desired.

$$u_{\text{ref}} = \begin{cases} \sqrt{(U_{\text{ref}}^2 - v^2)} & |U_{\text{ref}}| > |v| \\ U_{\text{ref}} & else \end{cases} \tag{3.6}$$

The low-level controller is then provided the u_{ref}. Even as the controller in inputted a surge reference, it has effectively become a speed controller. The speed controller applied is a feedback PI-controller,

$$\delta_{\text{throttle}} = -(e_u K_{p,U} + \int e_u K_{i,U}), \tag{3.7}$$

where $e_u = u_{\text{ref}} - u$.

The course is controlled through the subtraction of sideslip from the course reference to create a heading reference. The rudder input is defined by the PD control law,

$$\delta_{\text{rudder}} = -(e_\psi K_{p,\chi} + (r_{\text{ref}} - r) K_{d,\chi}) \tag{3.8}$$

where, $e_\psi = \psi_{\text{ref}} - \psi$.

3.4 Path following and line of sight guidance

Even though in everyday speech the terms *trajectory tracking* and *path following* may seem to be equal terms, they are not. In trajectory tracking, the objective is to control the vessel to be at a particular position at a particular point in time. The reference values are often provided as a time-series which combines positions with time and allows the position to develop so that a route may be established.

Path following on the other hand only describes a route without the time-aspect associated with the positions. The objective is simply to minimize the distance to this path. One way of doing this is by commanding the course angle, to regulate the cross-track error to zero [26]. Should, say, a tailwind or ocean currents in the direction of travel be accompanying a vessel on its way from A to B it would be nice to allow the voyage to be conveyed faster. Whether or not one wishes to do so or to maintain a planned pace may depend on the mission. In this thesis *Line Of Sight* (LOS) guidance will be employed to perform path following in the creation of the nominal controller references.

The LOS scheme is a function that manipulates the course to be able to control the vessel towards its next waypoint. As stated in Assumption 5, the transition from one waypoint to the next is beyond the scope of this thesis.

The LOS guidance block uses the vehicle state $\boldsymbol{\eta}$, which contains both the position and orientation of the vessel, as input. The parameters used in the block are *lookahead* distance Δ.

The path-tangential angle, α_k, is calculated as

$$\alpha_k = tan^{-1}(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}), \tag{3.9}$$

and is a function of the position of the waypoint k, and the next waypoint $k + 1$. The path orientation angle α_k rotates the system from north - east coordinates in the NED-frame to a path fixed reference frame. In other words a frame whose 1. axis points in the direction from one waypoint to the next, and in which easily

describes key parameters for the current situation.

$$\boldsymbol{\epsilon}(t) = \mathbf{R}_p(\alpha)^T \cdot (\mathbf{p^n(t)} - \mathbf{p_k^n}) \tag{3.10a}$$

$$\boldsymbol{\epsilon}(t) = \begin{bmatrix} s(t) \\ e(t) \end{bmatrix} \tag{3.10b}$$

Where $s(t)$ is the along-track error, and $e(t)$ is the cross-track error.

$$\mathbf{R}_p(\alpha)^T = \begin{bmatrix} cos(\alpha_k) & -sin(\alpha_k) \\ sin(\alpha_k) & cos(\alpha_k) \end{bmatrix} \tag{3.11}$$

For this path following purpose, only the cross-track error is relevant. The course angle assignment (3.12a) consists of both the *path-tangential angle*, and the *velocity-path relative angle*.

$$\chi_d(e) = \chi_p + \chi_r(e) \tag{3.12a}$$

$$\chi_r(e) = tan^{-1}(\frac{-e}{\Delta}) \tag{3.12b}$$

$$\chi_p = \alpha_k \tag{3.12c}$$

The *lookahead* distance Δ is a tuning parameter which affects how steep the approach towards the path should be. Low values causes the convergence towards the straight-line path to be more drastic than higher values. A large Δ causes low χ_r which again causes slow convergence towards the path. A rule of thumb for larger ships is to choose Δ as 2-3 times the length of the vehicle [25]. Odin is a short and very maneuverable vessel, but still one that we wish to behave predictably and calmly in normal operation. The lookahead distance Δ has therefore been selected to be 100 meters. In Figure 3.4 the relationship between the parameters can be observed. The geometrical relationship $e(t)^2 + \Delta^2 = R^2$ demonstrates the relationship between the parameters in the guidance system.

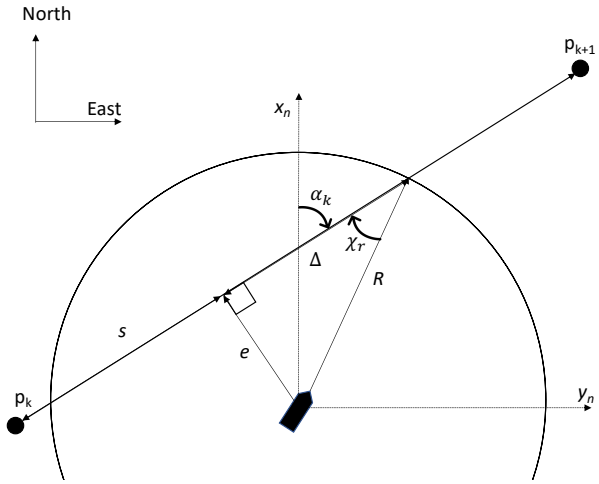


Figure 3.4: Lookahead Based Steering

In Figure 3.5 one can see an example trajectory of how a vessel may move when controlled by a LOS regime. As the vessel marked in green converges towards the center-line of cross-track error $e(t) = 0$, marked in purple, in the direction of the vector field of blue arrows, the path tangential angle decreases.

3.5 Sideslip compensation

Experiments have shown that Odin is a type of vessel that sideslips a substantial amount. This is due to the geometry of the hull and the combination of forces on the hull as the vessel maneuvers. This causes the difference between heading and course to become a critical factor in the navigation and control of this vessel. The heading is the direction of the bow of the boat, while the course is the direction of the velocity vector, which contains the underactuated sway component. The reason why the heading and course may differ originates from the sideslip angle.

There are two types of situations that could help us visualize sideslip. The first is in steady state travel where the vessel is affected by wind, current and waves. This could cause the vessel to travel in a slightly different direction than the heading. The difference between course and heading is denoted the sideslip angle. The

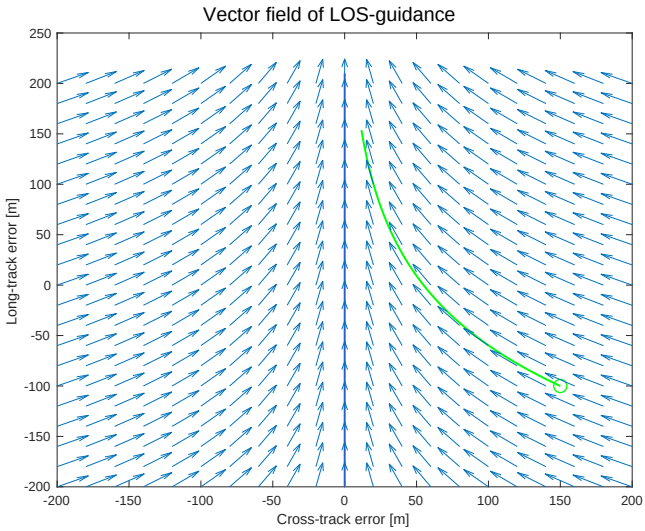


Figure 3.5: LOS-guidance vector field with an example trajectory

second situation may be in a course changing maneuver. As the vessel turns, its linear momentum must be shifted from its previous heading direction, to the new. In this process the speed is not solely based on surge anymore, but also sway. The sideslip angle is denoted β and the correlation between the vehicle course χ, heading ψ, surge and sway is given by Equation (3.13).

$$\chi = \psi + \beta \tag{3.13a}$$

$$\dot{\chi} = \dot{\psi} + \dot{\beta} \tag{3.13b}$$

$$\beta = atan\frac{v}{u} \tag{3.13c}$$

When evaluating the positions of the vehicle and the obstacles, the distinction between the vehicle's heading and course becomes critical.

In [7], Eriksen assumes that one can neglect sideslip in the trajectory generation by assuming $\psi = \chi$. In this thesis the sideslip has been taken into account to be able

to better model the vessels actual motions. This is done in the BC-MPC case, partly in the trajectory generation by using the actual course as the seeding direction of the tree and later subtracting the sideslip to achieve a dynamic heading reference which in practice is causing the PID-controller to be a course controller. In SB-MPC courses are evaluated, and the same conversion to heading is performed just before the signal is inputted the PID-controller.

3.6 Prediction of obstacle states

Both the BC-MPC and the SB-MPC algorithms employ linear prediction of obstacles. The position of obstacle i at time t is denoted $\boldsymbol{p}_i(t)$.

$$\boldsymbol{p}_i(t) = \begin{bmatrix} N_i(t) \\ E_i(t) \end{bmatrix} \tag{3.14}$$

Linear estimation of future states of obstacles are used, meaning that the current course and speed is assumed to be constant. This is commonly used in collision avoidance applications as it is a simple assumption which is not prone to noisy yaw-rate and acceleration estimates [15]. The assumption is fair as long as the prediction horizon is relatively short. The horizon of the predictions have been kept as long as the trees of the ownship's maneuvering possibilities. This allows for obstacle avoidance considerations for the duration of whole horizon.

The course of the obstacle is calculated as:

$$\chi_i(t) = atan2(\dot{E}_i(t), \dot{N}_i(t)) \tag{3.15}$$

which enables linear prediction of obstacle positions:

$$N_i(t) = N_i(t_0) + \dot{N}_i \cdot (t - t_0) \tag{3.16a}$$

$$E_i(t) = E_i(t_0) + \dot{E}_i \cdot (t - t_0) \tag{3.16b}$$

Where $N_i(t)$ and $E_i(t)$ denotes the estimated position of obstacle i at time t.

In the simulations performed in this thesis, the measurements are done at the same

time as the call, and the exact values for both velocities and positions are used to predict the future positions. This leads to the prediction of obstacle position being correct for all vessels that are traveling in straight line segments.

In the testing and verification of the BC-MPC algorithm Eriksen has performed full-scale experiments in the Trondheimsfjord in Norway. Moving obstacles has been observed by the test vessel Telemetron, which has been equipped with a radar based tracking and detection system. Odin has state of the art observation technology based on radar and lidar which one may assume will be able to perform at least at the same level as that of the Telemetron, resulting in less noise and well-founded control decisions.

Chapter 4

Branching-course MPC (BC-MPC)

In this chapter we will review the *branching-course* MPC algorithm. The BC-MPC algorithm was developed by Bjørn Olav Eriksen at NTNU as part of the *Autosea* project. It is a sample based MPC algorithm designed to be robust in regard to noisy obstacle estimates[7]. Eriksen states that the BC-MPC and other algorithms claiming to be *COLREGs-compliant*, are simply *COLREGs-aware*. The algorithm has been validated through both simulations and physical testing in the Trondheimsfjord in October 2017. An implementation limited to the course trajectory generation of the algorithm where performed as part of the specialization project [1]. This work has been brought into this thesis in the first parts of Chapter 4. This chapter will reuse and build on those results also presents further developments such as speed manipulation and an extensive cost function.

In [7], Eriksen develops "trees" of possible positions which are dependent on time based on a unified sampling of the possibility-space in both course and speed acceleration. As these are the two controlled modes in the 3 DOF model of the underactuated vessel this creates several alternatives which are all feasible for the vessel to perform. A helmsman of a *give way* vessel who encounters another vessel may resolve the situation by either altering his course, his velocity or both. That

the BC-MPC algorithm has the same possibilities will become clear as we now investigate the algorithm.

4.1 Vessel attitude control

In [7], the BC-MPC algorithm was implemented on a vessel with analytically known dynamics. A feedforward controller with feedback correction is therefore employed to control the vessel. This allows the vessel to follow smooth and differentiable references with little error.

Odin, however, has quite complex hydrodynamics, and a feedforward controller for this vessel does not yet exist. Such a control layer is not the focus of attention in this thesis and has therefore not been investigated further. We therefore employ feedback controllers. The end value of the course or speed is presented directly to the low-level controller as soon as the maneuver is to be initiated. The way to achieve accuracy in simulations is by selecting the tuning parameters of the tree generation to values which empirically causes the mathematically generated trajectories match well with those actually produced by the vessel model of Odin.

Feedforward control with feedback correction is probably a more accurate way of achieving coinciding trajectories than our feedback control with a step response in the reference passed to the controllers. However, the trajectory generation parameters have been tuned through simulations so that the error in the actual and projected trajectories are minimized.

4.2 Trajectory generation

The main steps of the trajectory generation part of the algorithm is reproduced from [7] in this section. This work was presented in the specialization project [1] as well, but the speed maneuver implementation is new for this thesis. Please note that some of the parameter names might be different from [7] to this thesis.

The variable N, which controls how many times the tree is branching was thoroughly investigated in the specialization project [1] in the fall of 2019. To have as many as 3 maneuvers in a sequence allowed for the vessel to plan its position

Table 4.1: Trajectory generation parameters for the BC-MPC algorithm

Description	Symbol	Typical values	Unit
Number of speed maneuver alternatives	N_U	$1-31$	
Number of course maneuver alternatives	N_χ	$1-31$	
Number of times a choice is made	N	$1-3$	
Length of timestep	T	$0.01-5$	s
Maneuver time	T_U and T_χ	$4-20$	s
Ramp time	T_{ramp}	1	s
The length of a single subtrajectory	T_{end}	$4-20$	s
Maximum speed acceleration	$\dot{U}_{\max}$	$0-1$	m/s^2
Maximum course acceleration	$\dot{r}_{\max}$	$0-\pi$	rad/s^2

in detail, while evaluating accurately its potential metrics for a fair horizon into the future. Eriksen concludes with 3 to be a nice value as well in [7], and suggests increasing the duration T_χ of each maneuver as a heuristic if one wishes to increase the overall horizon.

4.2.1 Trajectory generation: Single subtrajectory

The desired velocity trajectories are evenly spaced in the feasible search space for the desired vessel. In the proposed steps of the BC-MPC regime these values are achieved from the vessel model. This is because the limits of both course acceleration and speed acceleration are dependent on the vessel's current state. For simplification we have selected reasonable numerical values to be static in this implementation of the BC-MPC regime and this round of simulation.

The acceleration is limited as

$$\dot{U} \in \left[\dot{U}_{\min},\ \dot{U}_{\max}\right] \tag{4.1a}$$

$$\dot{r} \in \left[\dot{r}_{\min},\ \dot{r}_{\max}\right] \tag{4.1b}$$

There is created a uniformly distributed set of possible accelerations.

$$\dot{U}_{\text{samples}} = \{\dot{U}_1, \dot{U}_2, ..., \dot{U}_{N_U}\} \tag{4.2a}$$

$$\dot{r}_{\text{samples}} = \{\dot{r}_1, \dot{r}_2, ..., \dot{r}_{N_\chi}\} \tag{4.2b}$$

A measure for the steepness of a linear increase in acceleration is calculated.

$$k_{U,i} = \frac{\dot{U}_i}{T_{\text{ramp}}} \tag{4.3a}$$

$$k_{r,i} = \frac{\dot{r}_i}{T_{\text{ramp}}} \tag{4.3b}$$

This way of assigning values to the linear and angular acceleration is as described by Eriksen in [7]. Note that slight changes are made to the equations assigning acceleration rate from [7], although some typographical errors have been corrected as in the specialization thesis [1] and is stated in (4.4).

$$\dot{U}_{d,i}(t) = \begin{cases} k_{U,i} \cdot t & , 0 \leqslant t < T_{\text{ramp}} \\ U_{d,i}(t) & , T_{\text{ramp}} \leqslant< T_U - T_{\text{ramp}} \\ U_{d,i}(t) - k_{U,i} \cdot (t - (T_U - T_{\text{ramp}})) & , T_U - T_{\text{ramp}} \leqslant t < T_U \\ 0 & , T_U \leqslant t \leqslant T_{\text{end}} \end{cases} \tag{4.4a}$$

$$\dot{r}_{d,i}(t) = \begin{cases} k_{r,i} \cdot t & , 0 \leqslant t < T_{\text{ramp}} \\ 2\dot{r}_i - k_{r,i} \cdot t & , T_{\text{ramp}} < t \leqslant t < 2T_{\text{ramp}} \\ 0 & , 2T_{\text{ramp}} \leqslant t < T_\chi - 2T_{\text{ramp}} \\ -k_{r,i}(t - (T_\chi - 2T_{\text{ramp}})) & , T_\chi - 2T_{\text{ramp}} \leqslant t < T_\chi - T_{\text{ramp}} \\ -2\dot{r}_i + k_{r,i} \cdot (t - (T_\chi - 2T_{\text{ramp}})) & , T_\chi - T_{\text{ramp}} \leqslant t < T_\chi \\ 0 & , T_\chi \leqslant t \leqslant T_{\text{end}} \end{cases} \tag{4.4b}$$

When considering (4.4a) and (4.4b) analytically these constraints in the tuning of

the parameters must be satisfied,

$$T_{\text{ramp}} \cdot 2 \leqslant T_U \leqslant T_{\text{end}} \tag{4.5a}$$

$$T_{\text{ramp}} \cdot 4 \leqslant T_\chi \leqslant T_{\text{end}}, \tag{4.5b}$$

for them to produce reasonable output trajectories.

Numerical integration is performed to create sets of χ and U, i.e. sets containing *course* and *speed* subtrajectories for all the samples that is to be evaluated. It can be seen that the speed trajectories are continuously differentiable, while the course trajectories are twice continuously differentiable. The initial course rate is selected to be $r_{d,0} = 0$, to ensure continuous course profiles. The first maneuver trajectories are initiated with $r_{d,0} = r_0/2$ in the same way that the trajectories are initiated with current course and the previous value of the end of the speed trajectory maneuver. This proved to provide the best correspondence between actual trajectories and the predicted ones.

$$U_{d,i}(t) = U_{d,0} + \int_{t_0}^{t} \dot{U}_{d,i}(\gamma) d\gamma, \qquad i \in [1, N_U] \tag{4.6a}$$

$$r_{d,i}(t) = r_{d,0} + \int_{t_0}^{t} \dot{r}_{d,i}(\gamma) d\gamma, \qquad i \in [1, N_\chi] \tag{4.6b}$$

$$\chi_{d,i}(t) = \chi_{d,0} + \int_{t_0}^{t} r_{d,i}(\gamma) d\gamma, \qquad i \in [1, N_\chi] \tag{4.6c}$$

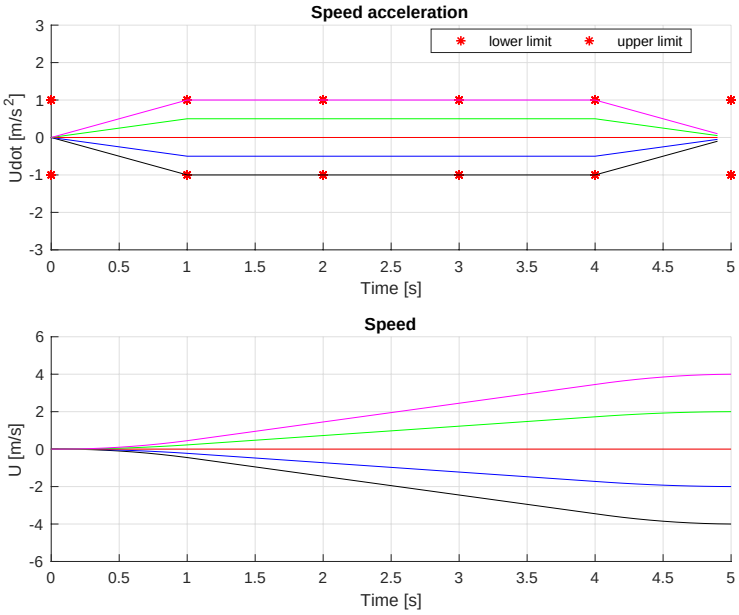


Figure 4.1: Speed - single subtrajectory

As described in (4.4a), the acceleration is altered linearly for T_{ramp} seconds until it reaches its desired level. It is kept at this value until T_{end} - T_{ramp}, then it begins to decent back towards zero linearly. The speed has then reached a new level in a smooth way. The resulting profiles may be seen in Figure 4.1

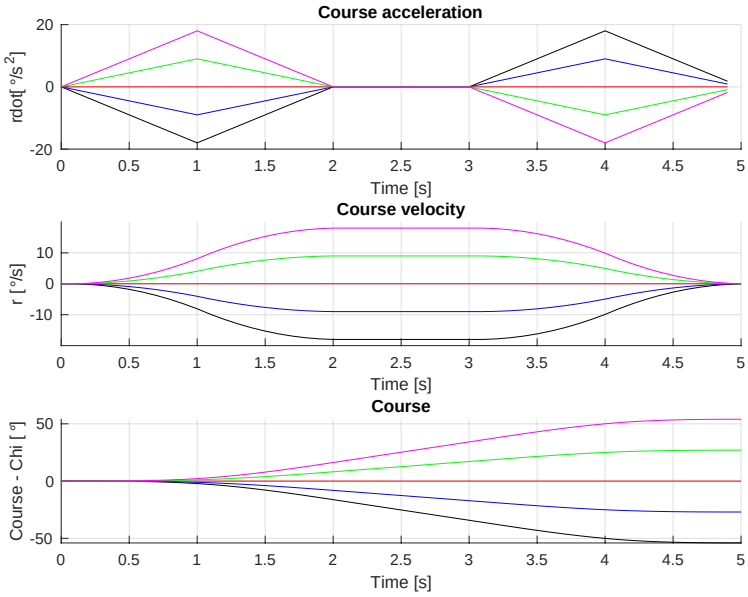


Figure 4.2: Course - single subtrajectory

Equation (4.4b) increases the course acceleration linearly from zero to its maximum value for the current sample before immediately descending to zero again, both actions each take T_{ramp} time. When the acceleration is kept constant, the course velocity is constant which evenly changes the course. Towards the end of the trajectory the opposite maneuver is performed by the course acceleration causing the velocity to return to zero and the course to stay constant. All three resulting profiles may be investigated in Figure 4.2

The two sets of desired velocity trajectories are combined to make a union set:

$$\mathcal{U}_d = \{U_{d,1}(t), U_{d,2}(t), ..., U_{d,N_U}(t)\} \times \{\chi_{d,1}(t), \chi_{d,2}(t), ..., \chi_{d,N_\chi}(t)\} \tag{4.7}$$

Here, Eriksen checks feasibility of the combinations using the vessel model. This is due to the fact that it may be a poor combination to turn as fast as possible while also accelerating as fast as possible. Using the vessel model to perform such a test has been beyond the scope of the BC-MPC implementation in this thesis.

Another difference in the implementations of the BC-MPC between Eriksen and this implementation is the feedback corrected pose trajectories.

$$\tilde{U} = \bar{U} - U_d \tag{4.8a}$$

$$\tilde{\chi} = \bar{\chi} - \chi_d \tag{4.8b}$$

Where in [7] Eriksen correlates:

$$\dot{\tilde{U}} = \frac{1}{T_{\tilde{U}}}\tilde{U} \tag{4.9a}$$

$$\dot{\tilde{\chi}} = \frac{1}{T_{\tilde{\chi}}}\tilde{\chi} \tag{4.9b}$$

In this thesis, the following simplification is made in line with Assumption 6:

$$\bar{U} = U_d \tag{4.10a}$$

$$\bar{\chi} = \chi_d \tag{4.10b}$$

4.2.2 Trajectory generation: Full trajectory

In this implementation of BC-MPC the speed is initialized by the last speed in the previous manoeuvre. This causes the speed reference to always be continuous. which is nice because it allows for evenly sized steps and the ability to return to the nominal value without several small steps which can be affected by other effects. The course trajectory however, is initialized from the course of the vessel itself. The current course rate is also used to create the first maneuver in the tree of maneuvers. The reason we found this best is because the tree will be as closely linked to the actual trajectories that would be achieved through selecting one of the candidates. This choice is possible because the vessel doesn't move until the calculation is ready in the simulated environment. However, in an actual implementation the vehicle movement during the computation time would need to be considered.

As a single subtrajectory is added onto the previous one to generate a tree, the last value in the previous subtrajectory is added to the new subtrajectory to generate a smooth multi-maneuver trajectory.

A tree of candidate solutions that is initialized from the last position, course and speed in the previous maneuver, would allow for the computation and evaluation of the tree to be performed on the vessel's way towards the trees root node. This would causes the system to be under the assumption that the controlled vessel is able to follow all provided references.

As the set of course and speed maneuvers $\mathcal{U}_d$ are produced, all the different combinations of these are different scenarios that our vessel could perform. Eriksen uses the vessel model and control allocation theory to check all combinations for feasibility. The vessel may for example, not be able to perform the maximum acceleration while at the same time performing the most extreme sequence of course maneuvers. Since we, in this thesis, do not perform such controls we have been forced to choose values for course acceleration and speed acceleration that are restricted enough so that all combinations that are available are fairly reasonable to assume that one would be able to follow. The cost of this is that the vessel is slightly restricted. When the vessel is not performing a course maneuver, it could in reality be able to accelerate even harder in speed, and vice versa. Unfortunately, this causes alternative actions that the boat *could* perform to never be in the set that is evaluated for selection. Eriksen's way of implementation is likely to achieve slightly better performance, and this is worth looking into in future work with the algorithm.

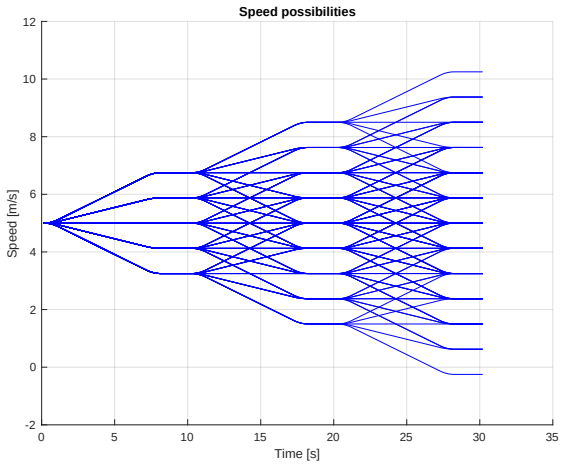


Figure 4.3: Speed possibilities in the set of speed maneuvers $N_U = 5$

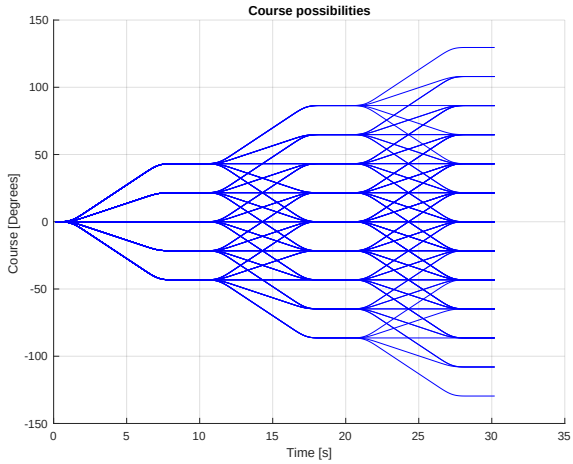


Figure 4.4: Course possibilities in the set of course maneuvers $N_\chi = 5$

In Figure 4.4 and 4.3 the speed and course possibilities in the set of maneuvers is plotted against time. The parameter $T_{\text{end}} = 10$ is set to 10 s, while T_χ and T_U is set to 8 s. Both N_U and N_χ is 5. $\dot{U}_{max} = 1/25\ m/s^2$ and $\dot{r}_{max} = pi/25\ rad/s^2$, while the

minimum limits are equal in magnitude with opposite signs.

4.2.3 Nominal alternative

In cases where there are no obstacles in the way, the vessel should follow its nominal path. Obstacles may have thrown the vessel of this course, and a LOS guidance law may be used to transition the vessel back to its intended path. The law as described in (3.12) is used to forward project the vessels position if it were to have this course for the duration of the horizon. However, no method for describing the transition from the vessels current pose to a pose where it is tracking this desired course χ_d has been implemented. This alternative that is added to the set of available trajectories is therefore not rooted in the vessels current pose, which may result in large steps in the course reference χ_{ref} from one iteration to another. This way of generating the trajectory and associated positions assumes an instantaneous turn, which is unrealistic. The difference in position and course from what is evaluated by the cost function and what is probable to occur as actual states will therefore be by far the largest of all the alternatives. If the LOS alternative is selected it may therefore be partly based on a weak basis. As the LOS course alternative is acting as the nominal alternative, this course trajectory is only available for combination with the nominal speed U_d.

4.2.4 Estimated scenario trajectory

Having achieved a set of velocities and courses that may be applied as references and applying Assumption 2, one may calculate velocities and positions in the NED frame. For each speed reference sequence, all the course reference sequences are applied.

$$\bar{\boldsymbol{p}}(t) = \begin{bmatrix} \bar{N}(t) \\ \bar{E}(t) \end{bmatrix} \tag{4.11a}$$

$$\bar{N}(t) = \int_{t_0}^{t} \bar{U}(\gamma) cos(\bar{\chi}(\gamma)) d\gamma \tag{4.11b}$$

$$\bar{E}(t) = \int_{t_0}^{t} \bar{U}(t) sin(\bar{\chi}(\gamma)) d\gamma \tag{4.11c}$$

The BC-MPC algorithm assumes that the controllers are able to follow the provided references, and for all combinations of course and speed maneuvers $\bar{\boldsymbol{\eta}}_k(t)$ is calculated. $\bar{\boldsymbol{\eta}}_k(t)$ is the estimated states of our vessel given a combination of maneuvers k. Since all predicted states belong to a scenario k the index has been omitted to ease the readability in the following sections.

In conventional 3 DOF modeling η is defined as $[N, E, \psi]^T$. However, Eriksen [7] uses χ as the last element in $\boldsymbol{\eta}$ due to the simplification of neglecting sideslip, hence assuming that the sideslip $\beta = 0$. A predicted scenario trajectory for the vessel then may be written as

$$\bar{\boldsymbol{\eta}}(t) = \begin{bmatrix} \bar{N}(t) \\ \bar{E}(t) \\ \bar{\chi}(t) \end{bmatrix}, \tag{4.12}$$

Unlike [7], we do not assume that $\chi = \psi$ and $\beta = 0$, and used the sideslip to calculate the actual course. We have kept the notation of Eriksen to allow for effective notation.

The predicted vessel pose trajectories are finally combined in the set $\bar{\mathcal{H}}$.

$$\bar{\mathcal{H}} = \{\bar{\eta}(t, \bar{U}(t), \bar{\chi}(t)) | \bar{U}(t), \bar{\chi}(t) \in \bar{\mathcal{U}}\} \tag{4.13}$$

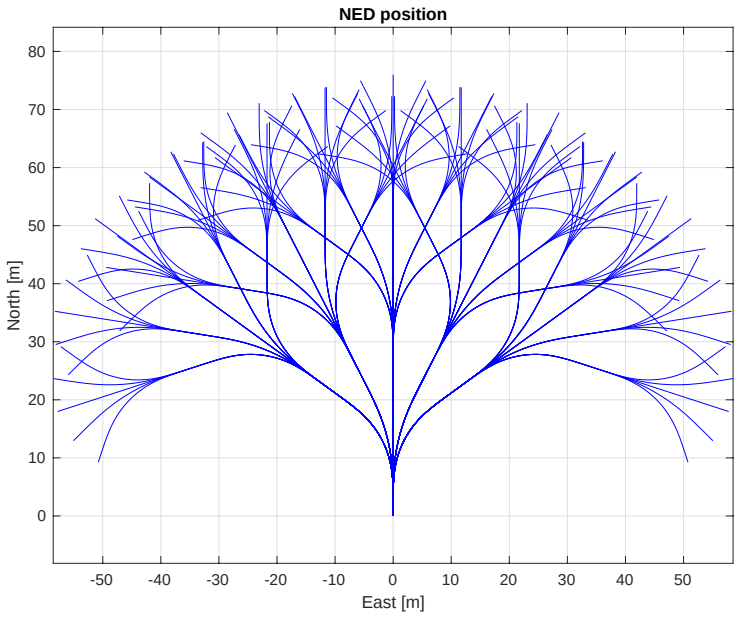


Figure 4.5: Subtrajectories combined to a search space of trajectories with $N = 3$, constant speed and $N_\chi = 5$

In Figure 4.5 one can see the subtrajectories being combined to form a set of feasible combinations of course acceleration, which positions are mapped out in. The parameter governing how many maneuvers should exist in a trajectory is here set to three, $N = 3$. N_χ, on the other hand is set to 5, which allows 5 different course rate profiles to travel by.

In Figure 4.6 the position trajectories of the set $\mathcal{H}$ can be seen. It includes all course trajectories tested for each speed trajectory. With $N = 3$, $N_\chi = 3$ and $N_U = 3$ this results in 27 course trajectories and 27 speed trajectories. The values of the parameters are selected for display purposes. When each of these are combined together this causes $N_\chi^N \cdot N_U^N = 729$ alternatives or scenarios that can be applied. As can be seen, are many of them fairly close to each other, but the differences between them and the way the room of opportunity is exploited is vastly increased compared to the case of fixed speed. These trajectories should cover the alternatives a real helmsman would evaluate and therefore avoid subsampling the continuous reality in the creation of alternative maneuver sequences. In the creation of this

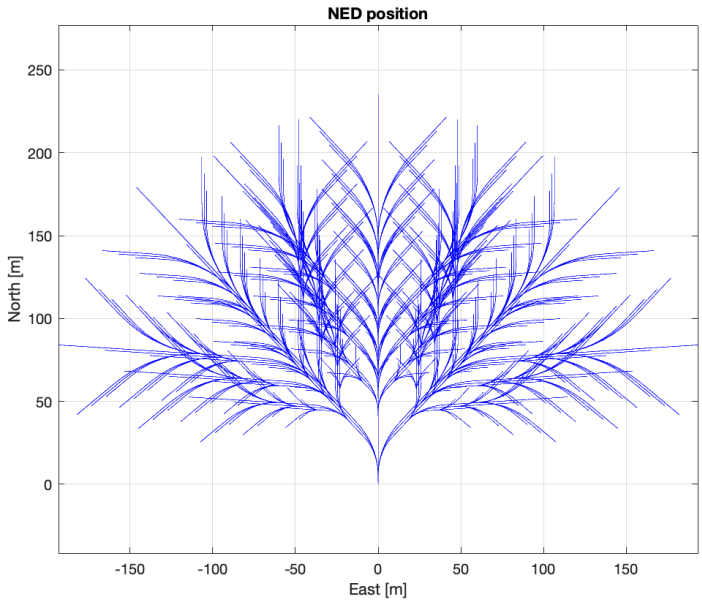


Figure 4.6: Subtrajectories combined to a search space of trajectories with $N = 3$, $N_U = 3$ and $N_\chi = 3$

figure, the max course acceleration applied is $\dot{r}_{\max} = \pi/25$ rad/s^2 and the max speed acceleration is $\dot{U}_{\max} = 1/25$ m/s^2.

4.3 Cost function

In this section Eriksen's cost functions from [7] is presented, with minor modifications. These modifications apply to the time varying vector $\boldsymbol{r_i}$ between an obstacle i, and the ownship. The align function has also been expanded with a speed deviance penalty. The tuning of all parameters have been based on the suggested values, but has been an area of exploration in Section 7. Some heuristics regarding time-varying weights have also been applied.

A cost function is a way of assigning a measure of optimality to an option. Optimization problems are typically formulated as a minimization problem where in this case, the desired trajectory is the one with the lowest cost function value. Such a value may be formed by assigning penalties, increase of value, for something

negative, and rewards, reduction of value, for something that is desired.

The trajectory that is minimizing the cost function is the combination of course and speed trajectories, χ^* and U^* which are passed to the vessel. Eriksen employs several functions for trajectory selection which take into account the heading and position of moving obstacles.

The optimization problem is defined by Eriksen in [7] is defined as

$$G(\bar{\boldsymbol{\eta}}_k(t), \boldsymbol{u_{d,k}}(t), \boldsymbol{p_d}(t)) = w_{al} \cdot align(\bar{\boldsymbol{\eta}}(t), \boldsymbol{p_d}(t)) + w_{av} \cdot avoid(\bar{\boldsymbol{\eta}}(t)) + w_t \cdot tran(\boldsymbol{u_d}(t)), \tag{4.14}$$

However, in this thesis, the cost function has been limited to

$$G(\bar{\boldsymbol{\eta}}_k(t), \boldsymbol{u_{d,k}}(t), \boldsymbol{p_d}(t)) = w_{al} \cdot align(\bar{\boldsymbol{\eta}}(t), \boldsymbol{p_d}(t)) + w_{av} \cdot avoid(\bar{\boldsymbol{\eta}}(t)), \tag{4.15}$$

although some discussion regarding the *transition* component of the algorithm is provided. The two remaining main components of the cost function is the *align* and the *avoid* functions which are presented in Section 4.3.1 and 4.3.2, respectively.

The optimal desired trajectory $\boldsymbol{u_d}^*(t)$ is consisting of the pair of optimal course and speed maneuvers. k is used to denote each scenario, as in each possible combination of a course and a speed maneuver sequence. Each scenario k, is evaluated by the cost function G, and the optimal input is found by selecting the one with the lowest cost function value.

$$\boldsymbol{u_d}^*(t) = \underset{(\bar{\boldsymbol{\eta}}_k(t), \boldsymbol{u_{d,k}}(t)) \in (\bar{\mathcal{H}}, \mathcal{U}_d)}{\arg\min} G(\bar{\boldsymbol{\eta}}_k(t), \boldsymbol{u_{d,k}}(t), \boldsymbol{p_d}(t)) \tag{4.16a}$$

$$\boldsymbol{u_d}^*(t) = \begin{bmatrix} \chi^*(t) \\ U^*(t) \end{bmatrix}, \tag{4.16b}$$

4.3.1 Align function

The *align* function has as its purpose to reward those trajectories in the tree maneuver sequences that correlate well with the nominal trajectory. It does so by penalizing the deviation from the desired forward projected LOS trajectory $\boldsymbol{p_d}(t)$.

$$\boldsymbol{p_d}(t) = \begin{bmatrix} N_d(t) \\ E_d(t) \end{bmatrix} \tag{4.17}$$

The desired course is calculated through the rate of change of the desired position.

$$\chi_d = atan2(\dot{E}_d, \dot{N}_d). \tag{4.18}$$

The trajectories are evaluated by the *align* function in (4.19).

$$align(\bar{\boldsymbol{\eta}}, \boldsymbol{p_d}(t)) = \int_{t=t_0}^{t_0+T_{\text{full}}} (w_p \left\| \begin{bmatrix} N(\gamma) \\ E(\gamma) \end{bmatrix} - \boldsymbol{p_d}(\gamma) \right\|_2 + w_\chi |\Upsilon(\bar{\chi}(\gamma) - \chi_d(\gamma))| + w_U |U(\gamma) - U_d| d\gamma \tag{4.19}$$

The *align* function compares the two trajectories in 3 ways over the duration of T_{full}. T_{full} represents the entire trajectory prediction time-horizon as is sums the duration of each subtrajectory N times.

$$T_{\text{full}} = \sum_{j=1}^{N} T_{\text{end}} \tag{4.20}$$

In the *align* function, first, the Euclidean distance is integrated over the horizon and scaled by w_p. Second, the sum of the differences between the predicted and desired course is calculated and scaled with w_χ. Υ is a function which maps an angle to the interval of $[-\pi, \pi)$. Last, the speed of a scenario is compared to that of the desired value U_d, this error is scaled by w_U.

4.3.2 Avoid function

The avoid function is the part of the BC-MPC's cost function which causes our vessel to avoid and keep distance to other obstacles. A vector, $\mathbf{r}_i(t)$, from the

ownship to obstacle i is employed.

$$\mathbf{r}_i(\bar{\boldsymbol{\eta}}(t); \boldsymbol{p}_i(t)) = \begin{bmatrix} \bar{N}(t) \\ \bar{E}(t) \end{bmatrix} - \boldsymbol{p}_i(t) \tag{4.21a}$$

$$\mathbf{r}_i(\bar{\boldsymbol{\eta}}(t); \boldsymbol{p}_i(t)) = \begin{bmatrix} r_{N,i} \\ r_{E,i} \end{bmatrix}, \tag{4.21b}$$

Please note that this vector is opposite of what is stated by Eriksen in [7], but the intent is the same, which is to create a vector from the obstacle to our ownship. The vector $\mathbf{r}_i(t)$ is used to calculate both the Euclidean distance between the obstacle and the ownship d_i, and the relative bearing from the obstacle to the ownship in relation to the obstacles' heading, β_i.

$$d_i(\bar{\boldsymbol{\eta}}(t), \boldsymbol{p}_i(t)) = |\mathbf{r}_i(d_i(\bar{\boldsymbol{\eta}}(t); \boldsymbol{p}_i(t)))| \tag{4.22a}$$

$$\beta_i(\bar{\boldsymbol{\eta}}(t), \mathbf{p}_i(t)) = \mathrm{Y}(\mathrm{atan2}(\mathrm{r_{E,i}}(\bar{\boldsymbol{\eta}}(\mathrm{t})), \mathrm{r_{N,i}}(\bar{\boldsymbol{\eta}}(\mathrm{t}))) - \chi_i(\mathrm{t})) \tag{4.22b}$$

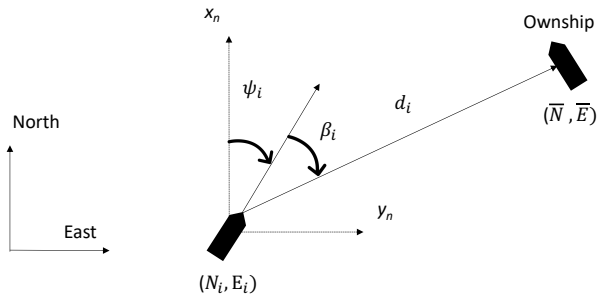


Figure 4.7: Distance d_i and relative bearing β_i between vessels. Figure inspired from [7].

The *avoid* function penalizes trajectories associated with risk of collision. It is built

up by integrating the weighted penalty over time for the whole horizon.

$$avoid(\bar{\boldsymbol{\eta}}(t)) = \sum_{i=1}^{M} \int_{t=t_0}^{t_0+T_{\text{full}}} w_i(\gamma) penalty_i(\bar{\boldsymbol{\eta}}(\gamma)) d\gamma, \tag{4.23}$$

where M is the number of obstacles.

While Eriksen [7] keeps the weights constant throughout the horizon, we have chosen to linearly decrease the weight of penalty to help the *avoid* function prioritize the dangers that appear closer and sooner to be weighted more, than distant more uncertain dangers.

$$\boldsymbol{w_i}(t) = 1.5 - \frac{t}{horizon} \tag{4.24}$$

The weight $w_i(t)$ applies for all obstacles, i. A variation where larger more inmaneuverable vessels or vessels with larger velocities where weighted heavier could also be relevant.

4.3.3 Penalty function

The *penalty* function penalizes the position of a time-instant in a trajectory by evaluating either only the distance d_i in the circular penalty case, or evaluating both the distance d_i and β_i together in the elliptical penalty case. The circular region is well suited for static obstacles, while the elliptical is better suited for moving obstacles.

Circular:

$$penalty_{i,circular}(\bar{\boldsymbol{\eta}}) = \begin{cases} 1 & , d_i < D_0 \\ 1 + \frac{\gamma-1}{D_1-D_0}(d_i - D0) & , D_0 \leqslant d_i < D1 \\ \gamma_1 - \frac{\gamma-1}{D_2-D_1}(d_i - D1) & , D_1 \leqslant d_i < D2 \\ 0 & , else \end{cases} \tag{4.25}$$

$\gamma_1 \in (0, 1)$ is a tuning parameter governing the penalty at distance D_1. Selecting $\gamma_1 = 0.1$ causes our vessel to be slightly influenced by the avoid function at distances larger than D_1, and to react much more resolute when $d_i(\bar{\boldsymbol{\eta}}(t), \boldsymbol{p}_i(t))$ is smaller than

D_1 and the penalty approaches 1. The circular penalty region can be seen in Figure 4.8.

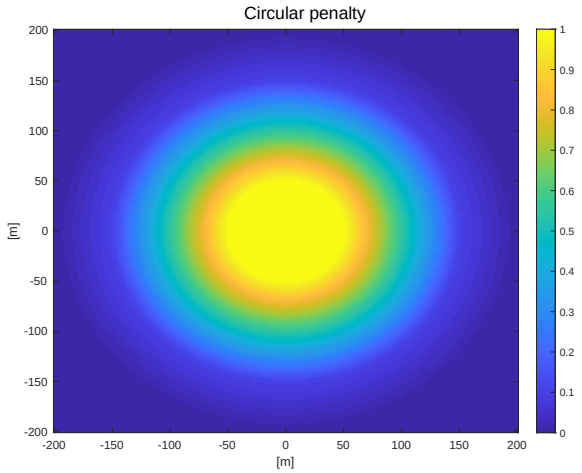


Figure 4.8: Circular penalty region

The elliptical COLREGs regions are created through the conversion to the circular parameters $D_k, k \in [0, 1, 2]$.

$$D_k(\beta_i) = \begin{cases} b_k & , \beta_i < -\frac{\pi}{2} \\ \frac{a_k b_k}{\sqrt{(b_k cos(\beta_i)^2 + (a_k sin(\beta_i)^2}} & -\frac{\pi}{2} \leqslant \beta_i < 0 \\ \frac{a_k c_k}{\sqrt{(c_k cos(\beta_i)^2 + (a_k sin(\beta_i)^2}} & 0 \leqslant \beta_i < \frac{\pi}{2} \\ \frac{b_k c_k}{\sqrt{(c_k cos(\beta_i)^2 + (b_k sin(\beta_i)^2}} & \frac{\pi}{2} \leqslant \beta_i \end{cases} \tag{4.26}$$

$$\boldsymbol{a} = \begin{bmatrix} a_0, & a_1, & a_2 \end{bmatrix} \tag{4.27a}$$

$$\boldsymbol{b} = \begin{bmatrix} b_0, & b_1, & b_2 \end{bmatrix} \tag{4.27b}$$

$$\boldsymbol{c} = \boldsymbol{b} + d_{COLREGs} \tag{4.27c}$$

The values in vector $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ determine the parameters of the elliptic penalty region surrounding the obstacles. These may be tuned to change our vessels actions in regard to the positions around obstacles. The elliptical penalty region can be seen in Figure 4.9.

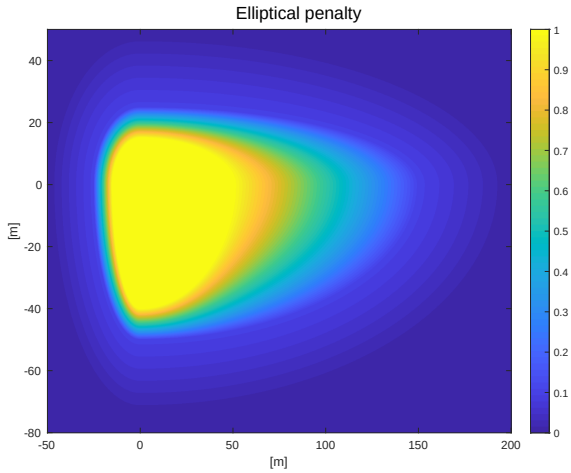


Figure 4.9: Elliptical penalty region. Notice the axis in regards to the obstacle position (0,0)

4.3.4 Inner penalty

The inner penalty is a subfunction of the penalty function which has its purpose in penalizing positions inside D_0 or the area described by a_0, b_0 and c_0 in the elliptical penalty function is applied. The reason one would want to do this is to further penalize positions which are inside the collision region. It is fairly logical that being in the center of the collision region is a worse situation than being at its edge. However, the penalty is determined by a situational parameter y_b which is not trivial to calculate. Our experience is that the controlled vessel steers well clear of the danger region even without this function, and it seems somewhat superfluous as the algorithm performs adequately and achieves reasonable results without it. It has therefore not been implemented.

4.3.5 Transition function

The *Transition* part of the cost function has the function of awarding the scenarios that are closest related to the previous selected scenario. This causes the overall actions of the vessel to be more consistent and less subject to noisy obstacle estimates. However, if a drastically different scenario (or combination of maneuvers) is evaluated by the other parts of the cost function to be better, then it will be selected. This is the result careful tuning of weight parameters affecting the magnitude of the different parts' contribution to the overall cost.

In this thesis, the states of the obstacles are fully available for the algorithm to use, while in more complex simulations and real applications of such an algorithm these states would have to be estimated. The obstacles are traveling linearly along a predefined path, and the forward prediction of the obstacles as described in equation (3.16) are also linear. This is an assumption that is fair for short periods of time in many cases, but in this case, it is absolutely accurate. There are therefore no "surprises" for Odin that would cause it to drastically change its plan, and we have observed that the BC-MPC algorithm functions fine without this part of the algorithm in place. In practice, some sensor noise would have to be expected, but given the state-of-the-art object detection and tracking systems currently on Odin the problem is assumed to be manageable.

Chapter 5

Sample Based MPC (SB-MPC)

In this chapter the SB-MPC algorithm is reviewed. It was originally presented in [15]. Inger Berge Hagen wrote her master's thesis [14] at NTNU in 2018 and explored in greater detail the SB-MPC and also contributed in publishing [16]. Preliminary full-scale testing was also attempted, but not completed. In 2019 the algorithm was part of Dutch initiative for sea trials of collision avoidance algorithms [27], and the approach was shown to be capable of finding and performing safe maneuvers in challenging situations.

5.1 Introduction to SB-MPC

A similarity between the way the SB-MPC and BC-MPC approaches are built up is the way trajectories are generated based on the input from a higher-level planner. The BC-MPC can be said to generate a tree of alternative paths, where the chosen path is the one which best both avoids obstacles and correlates to the nominal planned trajectory. If there are no obstacles, the higher-level alternative is selected. The SB-MPC algorithm's trees are directed in the direction of χ_d from the higher-level planner or a nominal guidance law. In the case of this thesis, the LOS guidance law is used. The tree consists of straight line segments, which each

has a constant offset to the nominal course χ_d. The limits for width of offsets has been set to $[-90°, 90°]$. This allows for the collision avoidance layer to simply add an offset, while at the same time continuing towards its long-term goal. The SB-MPC evaluates several speed coefficients in relation to each course offset as well. The output from the SB-MPC algorithm is χ_{ca} and P_{ca}, a course offset and a speed coefficient, respectively. These are employed to create the speed and course references as can be seen in (5.1).

$$U_{\text{ref}} = U_d \cdot P_{\text{ca}} \tag{5.1a}$$

$$\chi_{\text{ref}} = \chi_d + \chi_{\text{ca}} \tag{5.1b}$$

If no obstacles occur, then the SB-MPC outputs a zero course offset and a speed coefficient equal to 1, which causes the vessel to simply follow the LOS-regime with $U_{\text{ref}} = U_d$ and $\chi_{\text{ref}} = \chi_d$. However, if there are obstacles in the way, the SB-MPC provides a combination of a course offset χ_{ca} and a speed factor P_{ca} to manipulate the new references. The references provided to the lower level controllers are then a new course reference and a new speed reference.

5.2 Trajectory generation

The way the course reference is generated is different in the two algorithms. The BC-MPC approach creates a smooth three with multiple turns for each trajectory candidate, while the SB-MPC simply draws straight line segments from the vessels current position. One of the differences between the two - are in the direction of the tree. SB-MPC is directing the center of the tree in the χ_d-direction, meaning, the direction recommended by the higher-level planner which does not, in our case, take other vessels or obstacles into consideration in its planning. BC-MPC accepts some error in position and course from what is being evaluated by the cost function to do this. BC-MPC is directing its tree in the direction of its current course. All of the trajectories in the BC-MPC tree are built on the basis that they shall be as close to the actual trajectory that would be generated if the current scenario is chosen, and therefore starts with the vessel pose.

The SB-MPC algorithm has several similarities to the BC-MPC. Some of the param-

eters are thereby similar and are named equally in this thesis. For the parameters that only exist in one of the methods the naming conventions applied in the originating papers are kept and applied. Parameters for trajectory generation along with suggested value ranges are presented in Table 5.1.

Table 5.1: Trajectory generation parameters for the SB-MPC algorithm

Description	Symbol	Typical values	Unit
Number of speed maneuvers	N_U	1-10	
Number of course maneuvers	N_χ	1-31	
Course offset limits		[-180,180]	°
Number of times a choice is made	N	1	
Length of timestep	T	0.01-5	s
Maneuver time	T_U and T_χ	0	s
Horizon	T_{full}	30-300	s
The length of a single subtrajectory	T_{end}	30-300	s

The SB-MPC algorithm is spanning its tree from the vessels current position and in the direction of χ_d with 13 straight-line course trajectories for each speed alternative.

- Course offset (χ_{ca})[°]: [-90, -75, -60, -45, -30, -15, 0, 15, 30, 45, 60, 75, 90].
- Speed factor (P_{ca}): [1.0, 0.5, 0.0]

The course offset vector is containing each of the course offset alternatives which are referred to as χ_{ca}. The speed factor vector is similarly built up with the elements denoted as P_{ca}. In the candidate trajectory, the desired course χ_d is kept constant for the whole horizon, while in reality χ_d would change as the vessel approaches the straight-line path, allowing smooth convergence. The trajectories are denoted k to correspond with a combination of speed coefficient and course offset in the same way as in the BC-MPC algorithm.

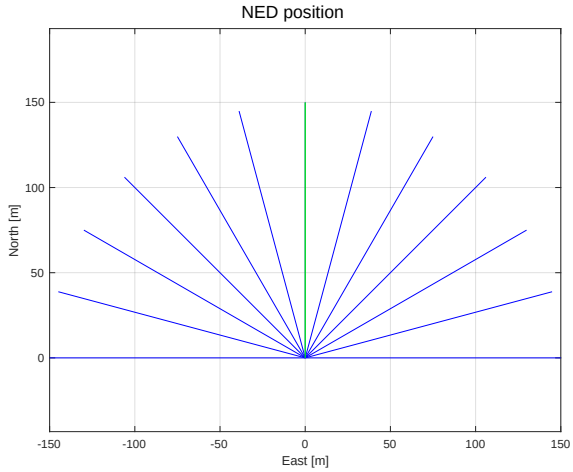


Figure 5.1: SB-MPC trajectory tree with predicted trajectories for nominal speed in each course alternative

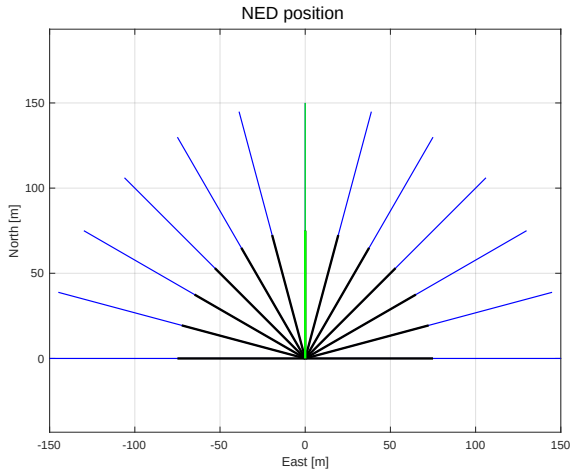


Figure 5.2: SB-MPC Position trajectory tree with all three speed alternatives for each course alternative

The blue lines in Figure 5.1 and 5.2 correspond to the alternatives from the nominal speed, $P_{\text{ca}} = 1$. The green lines indicate the trajectory for the zero course offset

trajectory. The black lines correspond to the trajectories of the vessel at half nominal speed $P_{\text{ca}} = 0.5$. The last speed alternative $P_{\text{ca}} = 0.0$ does not show in the figure, but is represented by a constant position for the whole horizon. The predictions are all made with the assumption of instantaneous turn, and instantaneous speed change.

5.3 Cost function

In this section the SB-MPC cost function from [14] is presented, with minor modifications. The obstacle positions in combination with the generated vessel scenario trajectories lays the foundation for the calculation and notation of the parameters in Table 5.2.

Table 5.2: Notation for the SB-MPC algorithm

Description	Symbol
Predicted velocity of the USV at a future time instant t, in scenario k.	$\vec{v}_0^k(t)$
Predicted velocity of obstacle i at a future time instant t, in scenario k.	$\vec{v}_{0,\text{i}}^k(t)$
Predicted distance to obstacle i at time instant t, in scenario k.	$d_{0,\text{i}}^k(t)$
The distance within which the COLREGs is said to apply for the obstacles.	d_{close}

The indexing used are

$$k \in \{1, 2, ..., N_s\} \tag{5.2a}$$

$$i \in \{1, 2, ..., N_o\}, \tag{5.2b}$$

where the index k denotes the scenario, while i denotes the obstacle.

Each scenario is evaluated by a cost function consisting of several components. These components are summed together in the overall *Hazard* function. It consists of an obstacle-specific part, two parts which penalizes deviation from the previous elected control behavior and two parts which penalize deviation from the desired control behavior. The obstacle specific part consists of the product of risk and cost, and the COLREGs binary indicator $\mu_i^k(t)$ which is scaled by the weight κ_i.

These components are summed together in the Hazard function:

$$\begin{aligned}\mathcal{H}^k(t_0) = \max_i \max_{\mathrm{t}\in\mathrm{D}(\mathrm{t}_0)} \left(C_i^k(t)R_i^k(t) + \kappa_i \mu_i^k(t)\right) \\ + \Delta_\chi(\chi_{\mathrm{ca}}, \chi_{\mathrm{ca,last}}) + \Delta_P(P_{\mathrm{ca}}, P_{\mathrm{ca,last}}) + K_P(1-P) + K_\chi \chi_{\mathrm{ca}}^2\end{aligned} \tag{5.3}$$

Here, the function R denotes the risk and is described in Section 5.3.1. The function C defines the cost of collision and is described in Section 5.3.2. The cost of deviance of course and speed, and the COLREGs compliance operator is described in Sections 5.3.3 and 5.3.4. $D(t_0) = \{t_0, t_0 + T_s, ..., t_0 + T\}$ where T_s is the discretization interval of 0.1 seconds and T is the prediction horizon of the MPC. $T = 45$ seconds is applied.

The optimal control control input is the one associated with scenario $k^*(t_0)$ which is the one that minimizes the cost function.

$$k^*(t_0) = \arg\min_k \mathcal{H}^k(t_0) \tag{5.4}$$

5.3.1 Risk of collision

In many applications such as insurance calculations, failure mode analysis in engineering applications and project management disciplines, the *risk* of an event is estimated through the multiplication of *probability* of that event occurring and magnitude of the resulting *impact*. In [14], the way risk is evaluated is by letting *Danger* be the product of *Risk of collision* and *Cost*. The concept is the same and creates a simple way of weighing both the likelihood and the severity of a danger. The function for risk of collision is

$$R_i^k(t) = \begin{cases} \frac{1}{|t-t_0|^p}\left(\frac{d_{\mathrm{safe}}}{d_{0,i}^k(t)}\right)^q & \text{, if } \mathrm{d}_{0,\mathrm{i}}^{\mathrm{k}}(\mathrm{t}) < \mathrm{d}_{\mathrm{safe}} \\ 0 & \text{, otherwise} \end{cases} \tag{5.5}$$

where the risk is functioning as a likelihood-operator. It is predicted as a function of time and predicted distance from an obstacle. The prediction of risk must be at a time $t > t_0$. The way the risk is calculated can be tuned through the parameters d_{safe}, p and q. d_{safe} establishes the boundary for where one at the time of the call evaluates the risk of collision with obstacle i. If the distance at the time of the function call is smaller than d_i^{safe}, the parameter works as a scaling parameter

where large values cause larger risk for the same distance. The parameter p is the exponent weighing the distance in time until risk occurs, while $q \geq 1$ weighs the distance at a time in relation to d_i^{safe} exponentially. Hagen states that these parameters are the key parameters for Rule 16 compliance [14], which is the duty of the give-way vessel to keep well clear of all other vessels and do so in due time.

5.3.2 Cost of collision

The cost of collision is calculated in [14] as the difference in the two involved vessels' velocity squared, multiplied with a scalar gain K_{coll}. This gain could be dependent on the size and type of the obstacle or the position in regard to the velocities, but is in this implementation a constant value. The form of the function reminds of that of kinetic energy $E = 1/2m * v^2$ which is not a bad measure for the severity of a collision.

$$C_i^k(t) = K_{\text{coll}}(t) \left| \vec{v}_0^k(t) - \vec{v}_{\text{i}}^k(t) \right|^2 \tag{5.6}$$

If the ownship where to find itself very close to an obstacle, with quite similar course and speed, the calculated *cost of collision* would be very small. However, in reality, all collisions are bad, and the collisions that result in a large spill of kinetic energy are of course correspondingly worse. Penalizing all collisions with a base cost, in addition to the afore mentioned velocity dependent part of the cost might help avoid situations collisions where both vessels are traveling in the same direction at similar speeds not far from each other.

$$C_i^k(t) = K_{\text{coll}} \left| \vec{v}_0^k(t) - \vec{v}_{\text{i}}^k(t) \right|^2 + K_{\text{coll}} C_{\text{base}} \tag{5.7}$$

Equation (5.7) has been applied with success in the simulations in this thesis. Another way of influencing the calculation of the cost of collision could be to use the rate of change in the length of the LOS vector L. If this is decreasing rapidly, then the two vessels are approaching each other. In the case of two vessels driving away from each other they have a large difference in their velocities, causing a large *cost of collision*. Unfortunately, this penalty occurs even though it is desirable for the vessels to increase their distance, if they wish to avoid collision.

5.3.3 Cost of course and speed deviance

The subfunction $K_P(1 - P_{\text{ca}})$ causes a penalty to all speed coefficients that are not equal to 1. The function is built to take elements $P_{\text{ca}} \in [0, 1]$. The deviation from nominal course is penalized by $K_\chi \chi_{\text{ca}}^2$, causing course offsets to be penalized quadratically.

The COLREGs rule 8 state that it is desirable to maneuver in a predictable way, and one therefore wishes to minimize the change of course and speed over time. At the same time, one does not wish to do so at the cost of safety. A penalty for changing the course and speed from one iteration to the next is implemented. It is important that this does not exceed the magnitude of the danger part of the cost function, as that should always be prioritized [16].

The cost of course change is calculated as

$$\Delta_\chi = \begin{cases} K_{\Delta\chi,\text{port}}(\chi_{\text{ca}} - \chi_{\text{ca-last}})^2 & \text{, if turn to port} \\ K_{\Delta\chi,\text{starboard}}(\chi_{\text{ca}} - \chi_{\text{ca-last}})^2 & \text{, if turn to starboard} \end{cases} \tag{5.8}$$

The reason that there are two different values for the gains for the penalty to starboard and to port is due to the wish of changing our course to starboard in the event of a *head-on* situation (rule 14) or *crossing* situation (rule 15). Therefore, the relation; $K_{\Delta\chi,\text{port}} < K_{\Delta\chi,\text{starboard}}$ should apply.

In the case of change of speed factor, the function is as follows:

$$\Delta_P = K_{\Delta\text{P}} \, |P_{\text{ca}} - P_{\text{ca-last}}| \tag{5.9}$$

The previous commanded propulsion $P_{\text{ca-last}}$ is compared to the candidate P_{ca} of scenario k and deviations are penalized linearly causing an effect of consistency in speed changes.

5.3.4 COLREGs compliance

To achieve control inputs which not only minimize danger, but also complies with COLREGs, a binary operator $\mu_i^k(t)$ is implemented. It evaluates if our ownship, with the selection of the control input associated with scenario k is breaking any rules in its relation to obstacle i at time t. This operator is triggered to be true,

$\mu_i^k(t) = 1$ when certain evaluation criteria is fulfilled. The LOS unit vector from our vessel to obstacle i is denoted $L_i^k(t)$. This is much like, but in the opposite direction of the $r_i(t)$-vector from the BC-MPC scheme. In Table 5.3 the angles used for classification is presented.

Table 5.3: Classification parameters for the SB-MPC algorithm

Parameter	Value	Description
ϕ_{ot}	68.5°	Overtaking
ϕ_{ho}	22.5°	Head-on
ϕ_{ah}	15.0°	Ahead
ϕ_{cr}	68.5°	Crossing

$$\mu_i^k(t) = \text{RULE14 or RULE15} \tag{5.10a}$$

$$\text{RULE14} = \text{CLOSE \& STARBOARD \& HEAD} - \text{ON} \tag{5.10b}$$

$$\text{RULE15} = \text{CLOSE \& STARBOARD \& CROSSING \& NOT OVERTAKEN} \tag{5.10c}$$

In other words, $\mu_i^k(t)$ is true, when either Rule 14 or 15 is broken. The terms for their individual triggering can be seen in Eq. (5.10) b and c.

CLOSE An obstacle i is CLOSE when it is closer than the threshold distance d_{close}.

$$d_{0,\text{i}}^k \leq d_{close} \tag{5.11a}$$

OVERTAKEN A vessel is classified as OVERTAKEN if it is CLOSE, the obstacle speed is larger than our ownship speed, and the angular difference in the velocities are less than ϕ_{ot}. The requirements are:

$$d_{0,\text{i}}^k \leq d_{close} \tag{5.12a}$$

$$|\vec{v}_i(t)| > |\vec{v}_0^k(t)| \tag{5.12b}$$

$$\vec{v}_0^k(t) \cdot \vec{v}_i(t) > cos(\phi_{\text{ot}}) \left|\vec{v}_0^k(t)\right| |\vec{v}_i(t)| \tag{5.12c}$$

STARBOARD Obstacle i is said to be STARBOARD of our ownship if the bearing angle of $\vec{L}_i^k(t)$ is larger than the heading angle of our ownship.

HEAD-ON An obstacle can be said to be *head-on* if it is CLOSE, and

$$|\vec{v}_i(t)| > 0.05 \tag{5.13a}$$

$$\vec{v}_i(t) \cdot \vec{v}_0^k(t) < -cos(\phi_{\text{ho}})|\vec{v}_i(t)||\vec{v}_0^k(t)| \tag{5.13b}$$

$$\vec{v}_0^k(t) \cdot \vec{L}_i^k(t) > cos(\phi_{\text{ah}})|\vec{v}_0^k(t)| \tag{5.13c}$$

Where the first equation translates to that the obstacle is moving. The second that their courses are towards each other with an error less than $\phi_{\text{ho}} = 22.5°$. And the last equation that the speed of the ownship is in the direction of the obstacle within a $\phi_{\text{ah}} = 15°$ margin.

CROSSING For the vessels to be defined as CROSSING, they need to fulfill

$$\vec{v}_i(t) \cdot \vec{v}_0^k(t) < cos(\phi_{\text{cr}})|\vec{v}_i(t)||\vec{v}_0^k(t)|, \tag{5.14}$$

which means that their velocities have a larger angular difference than $\phi_{\text{cr}} = 68.5°$.

5.3.5 COLREGs transitional cost

The $\mathcal{T}_i^k(t)$-operator is a suggested addition to the cost function which is proposed in [16]. The $\mathcal{H}^k(t_0)$ is as before only now with the addition of $\mathcal{T}_i^k(t)*_i$, where the first is a binary operator $\in \{0, 1\}$ and the latter is the tuning parameter acting as weighting in regards to the overall function. The purpose of this part of the function is to prevent our vessel to pass an obstacle in another way than originally intended by the current control behavior.

By comparison this part of the cost function has a lot of the same properties and functions as the *Transition* part of the BC-MPC cost function. For the same reasons that the *Transition* part of the BC-MPC cost function was left out, the COLREGs transitional cost $\mathcal{T}_i^k(t)$, is left out of the SB-MPC regime. This allows for the algorithms to compete on equal terms and as they are not exposed to faulty sensor information and only encounter straight-line traveling, constant velocity obstacles they should both be able to perform well. Yet, the COLREGs transitional

cost operator is still presented here:

$$\mathcal{T}_i^k(t) = O_i^k(t) \vee Q_i^k(t) \vee \mathcal{X}_i^k(t), \tag{5.15}$$

where the following binary operators are applied:
$O_i^k(t)$ = 1 - The USV is overtaking vessel i in scenario k at time t.
$Q_i^k(t)$ = 1 - The USV is being overtaken by vessel i in scenario k at time t.
$\mathcal{X}_i^k(t)$ = 1 - The USV is crossing paths with vessel i in scenario k at time t.

Overtaking: The overtaking indicator is triggered in by the following logical relation:

$$O_i^k(t) = \begin{cases} O_i(t_0) \wedge \mathcal{S}_i^k(t) & , if \neg \mathcal{S}_i(t_0) \\ O_i(t_0) \wedge \neg \mathcal{S}_i^k(t) & , if \mathcal{S}_i(t_0) \end{cases}, \tag{5.16}$$

where, $\mathcal{S}_i(t_0)$ indicates that the obstacle i is currently on the starboard side of Odin, $\mathcal{S}_i^k(t)$ indicates that the obstacle i at time t in scenario k is starboard of our vessel.

The first case in the overtaking criterion at time t is that of the obstacle is not starboard at the time of call, but starboard and overtaking at the time t later on. The latter one is triggered if the vessel is overtaking at a point in time if it is both overtaking and starboard at the time of call, t_0, but not not starboard at the time t.

When $O_i(t_0)$ = 1,Odin is currently overtaking vessel i if the obstacle is considered *close ahead* and is traveling at a *lower speed.* As long as the obstacle's speed is not close to zero, this condition also applies.

$$\vec{v}(t_0) \cdot \vec{v}_i(t_0) > cos(\phi_{\text{ot}}) \, |\vec{v}(t_0)| \, |\vec{v}_i(t_0)| \tag{5.17}$$

Overtaken: The overtaking operator $Q_i^k(t)$ is simply evaluating the situation from the perspective of the obstacle. If the obstacle satisfying the criterion for *Overtaking*, then our vessel is classified as a vessel which is being overtaken and $Q_i^k(t)$ = 1.

Crossing:

$$\mathcal{X}_i^k(t) = X_i^k(t_0) \wedge \mathcal{S}_i(t_0) \wedge \mathcal{S}_i^k(t) \wedge \text{turn to port} \tag{5.18}$$

The USV is in a crossing situation if $\mathcal{X}_i^k(t) = 1$ with obstacle i if the obstacle is ahead and:

$$\vec{v}(t_0) \cdot \vec{v}_i(t_0) < cos(\phi_{\text{cr}}) \, |\vec{v}(t_0)| \, |\vec{v}_i(t_0)| \wedge \neg \mathcal{O}_i^k(t) \wedge \neg \mathcal{Q}_i^k(t) \tag{5.19}$$

The angle between the velocities are of a certain difference ϕ_{cr} while the vessels are not overtaking nor crossing.

One can see that the classifying criteria in the *COLREGs compliance* section differ slightly from that of the *COLREGs transitional cost*. The differences may originate from different the articles' - [14] and [16], implementation and investigation of the cost function. But the COLREGs transitional operator is investigating the development of a situation from t_0 to t whereas the first simply evaluates a moment in time t.

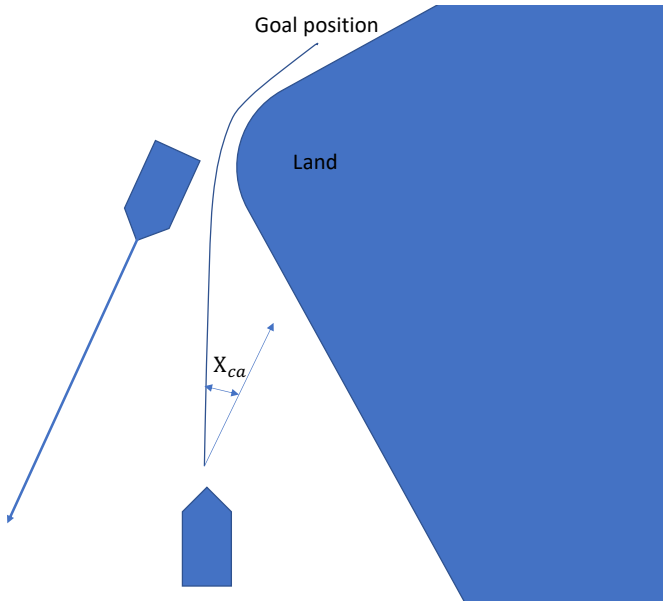


Figure 5.3: Collision avoidance situation with limitation in the surrounding waters

In the SB-MPC regime a grounding function g must be used to take into consideration the nature of the surrounding waters and to avoid selecting a χ_{ca} that could cause grounding. Such a situation can be seen in Figure 5.3. Simply using a higher-level planner that takes this into consideration and leaving it out of the low-level planner will be deficient.

Chapter 6

Implementation

In this chapter we will show how we have connected the BC-MPC and SB-MPC algorithms to the Odin model and its controllers. The tuning of each algorithm is also presented. The BC-MPC and SB-MPC algorithms have both been implemented to take in a desired direction and desired speed. They have both been given the task of solving the motion planning problem for the Odin model by then outputting course and speed references. The algorithms have been faced with the same situations in regards to obstacles and positions and the results are presented in Chapter 7. Through the simulations they are given equal opportunities to perform and their resulting references have been passed to the vessel's controllers. The performance of the overall system are then subject of evaluation.

Matlab 2019a and Simulink have been the main tools for implementation. The MSS-toolbox for guidance, navigation and control has also been used with some of its functions [28]. A 3 degree of freedom model of the vessel, its controllers and hydrodynamics was received from FFI at project initiation. The Simulink model is running with a variable step solver. This allows the solver to chose how large steps it should take in time dependent on the rate of change of values in the systems. If the dynamic response of the system is significant, the step-size is reduced. If not, the steps are increased to reduce computational load. However - the Matlab-fcn containing the low-level collision avoidance algorithm is set to be called at a fixed step-iteration of 0.1 seconds. This means that the reference value for both the

speed and course reference will be updated with a frequency of 10 Hz, which is a reasonable rate for the vehicle dynamics of Odin.

6.1 Low-level controllers

The first step of an overall process to implement a collision avoidance algorithm will be to make sure that the vessel control is performing consistently well, and is able to track simple references. This can be done by tuning the low-level course and speed controllers to perform benchmark maneuvers such as turning, stopping and acceleration behaviors to achieve an accurate and responsive vessel.

In [1], the practicalities behind the sideslip compensation, the speed and course controller tuning is presented in greater detail. However, the relationship between speed and surge reference is developed as a part of this thesis and is presented in Chapter 3. Here, key elements of the controllers are revisited. The controller tuning is presented in Table 6.1.

Table 6.1: Low-level controller tuning

Description	Value
$K_{p,\chi}$	200
$K_{d,\chi}$	600
$K_{p,U}$	8
$K_{i,U}$	4

The sideslip is subtracted from the selected and optimal choice of course trajectory, thus creating a heading reference for the vessel model. The rudder input is defined by the PD control law,

$$\delta_{\text{rudder}} = -(e_\psi K_{p,\chi} + (r_{\text{ref}} - r)K_{d,\chi}) \tag{6.1}$$

where,

- $r_{\text{ref}} = 0$
- $e_\psi = \psi_{\text{ref}}$-$\psi$

The reason r_{ref} is kept at zero instead of its actual value is to avoid large spikes in the event of steps in the reference signal.

We recall that the speed controller applied is a feedback PI-controller,

$$\delta_{\text{throttle}} = -(e_u K_{p,U} + \int e_u K_{i,U}), \tag{6.2}$$

where $e_u = u_{\text{ref}} - u$.

6.2 General tuning considerations

In general, one may view the overall tuning process of the two algorithms to be fairly similar. In the tuning of the algorithm parameters the suggested values from the papers of Hagen and Eriksen has been used as a starting point. In this section some general guidelines for tuning of them both are given, before the specific tuning of BC-MPC and SB-MPC is presented in Section 6.3 and 6.4, respectively.

The tuning distances found in both algorithms affect the distances in which the position is penalized. In SB-MPC it is the d_{safe} and d_{close} while in the BC-MPC the vectors **a**, **b** and **c** are the key parameters. The parameter d_{close} in SB-MPC is fairly equivalent with D_0 in BC-MPC which is composed by the bearing angle β and the distances a_0, b_0 and c_0. These parameters are the limit for when the penalty really starts to increase dramatically due to the risk of collision. In general, the lower the distances, the closer and more dangerously the vessels are allowed to be each other, thus allowing path following to be fulfilled in a larger degree. If the distances are large, the distances to travel around an obstacle becomes large. In this case it is necessary that the time horizon of the algorithms is seen in relation to the speed to allow for the algorithm to see *beyond* the obstacle's penalty region and find a path which brings the ownship closer to its goal in a safe way. If the tuning distances are large, and the horizon too short, it is possible that our ownship would take a very large detour lasting for several iterations of the function call, as it lacks the ability to see the long term downside of turning and moving parallel to the obstacle. Such behavior is not only suboptimal, but also unpredictable for the other vessels to observe, which again could lead to dangerous situations.

The time between each call of the low-level collision avoidance algorithm, T_{call},

called $T_{\text{BC-MPC}}$ in the specialization project [1], is also interesting. For SB-MPC the time, $T_{\text{call,SB-MPC}} = 5s$, while in the BC-MPC algorithm the time $T_{\text{call,BC-MPC}} = 10s$. This is due to the BC-MPC being more computationally heavy, and the need to perform a maneuver before the next is applied. To investigate them both it would be fair to have a difference in how often that may be called in these simulations as well. The investigation of run-time has not been the focus for this thesis.

The collision avoidance algorithms has now been tuned for transport missions along a straight-line nominal trajectory. For missions with a higher degree of need to follow a certain path, the lookahead distance of the LOS-Guidance must be lowered, and the cost of speed reduction should be lowered, to allow for the algorithm to «wait out» occurring situations, instead of driving off with an alternative course at nominal speed.

6.3 Tuning of the BC-MPC algorithm

The simulation parameters used on the BC-MPC algorithm in this thesis is mainly built up of trajectory generation and cost function tuning parameters. These are presented in Table 6.2.

6.3.1 Tree generation

The goal is to build a tree of trajectories which the vessel is able to perform, and which to a large degree covers the span of possible actions to avoid to heavy subsampling. The tuning of the tree generation in the BC-MPC algorithm was well investigated in the specialization thesis from the fall of 2019 [1]. The parameters regarding number of course alternatives, number of times the tree is branching and the acceleration limit values are in line with those findings. The time horizon, however, has increased from 30 to 45 seconds. The main focus of tuning in this thesis is placed in the speed branching and cost function tuning. The key parameters from Odin used in the implementation can be found in Table 6.2.

A difference between this implementation and the one of Eriksen [7] is the way the trajectories are initialized. When initializing the tree of trajectory candidates, the first subtrajectory in all of them must be initialized with the actual course as obtained from equation (3.13a). This decides the direction the tree is put in initially. See Figure 6.1. The other subtrajectories begin with the values the previous

Table 6.2: Selected tuning parameters for the BC-MPC algorithm

Description	Symbol	Typical values	Unit
Number of speed maneuver alternatives	N_U	5	
Number of course maneuver alternatives	N_χ	5	
Number of times a choice is made	N	3	
Length of timestep	T	0.1	s
Maneuver time	T_U and T_χ	8	s
Ramp time	T_{ramp}	1	s
The length of a single subtrajectory	T_{end}	15	s
Maximum course acceleration	$\dot{r}_{\max}$	$\frac{\pi}{25}$	rad/s^2
Maximum speed acceleration	$\dot{U}_{\max}$	$\frac{1}{25}$	m/s^2
Position weight	w_p	1	
Course/angular error scaling weight	w_χ	100	
Speed error scaling weight	w_U	50	
Align weight	w_{al}	1	
Avoid weight	w_{av}	6000	
LOS Lookahead distance	Δ	100	m
Major axis parameters	$\boldsymbol{a}$	[50, 150, 250]	m
Minor axis parameters	$\boldsymbol{b}$	[12, 20, 50]	m
Starboard axis parameters	$\boldsymbol{c}$	[27, 35, 65]	m
COLREGs distance	$d_{COLREGs}$	15	m
Obstacle cost gradient parameter	γ_1	0.1	

subtrajectory ends. Eriksen initializes the trajectories with the last value from the previous trajectory causing no steps in the reference. When initializing with the previous value the vessel may propagate errors in position and course, if the control layer is unable to follow the references.

The way the first course subtrajectory develops is caused by numerical integration of the course rate from equation (3.13b). The first subtrajectory of the course rate is initialized with half of the vessels current value, while the remaining subtrajectories are initialized by the last value in the previous vector. The effect the initial course rate has on the course tree is to bend the first part of it as can be seen in Figure 6.2.

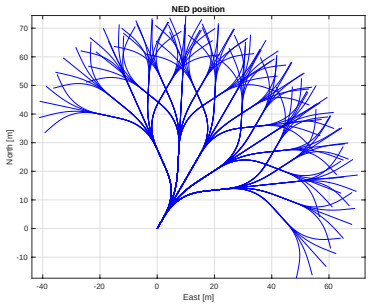


Figure 6.1: Course, $\chi = 30°$ at time of the BC-MPC call

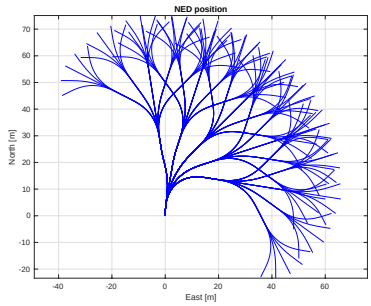


Figure 6.2: Course rate, $r = \pi/20$ rad/s at time of the BC-MPC call

As the tree of the BC-MPC algorithm is seeded in the direction of the vessel, it is possible for an incorrect tuning to cause circular behavior in the position of the vessel. If the tree is too radical in its steering it can cause a small area where the ownship may «spend time» being stuck in a local minima.

In the BC-MPC regime Eriksen also gives another reason to avoid sectioning the search space too fine which is to be able to have visible maneuvers. A counterargument to this, is that the difference between to alternatives in a tree may be small without this causing a problem, however a course or velocity change should be of such magnitude that it is easily observable.

6.3.2 Tuning of the cost function

In Section 4.3.3, the tuning vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ is presented. They span the penalty region around each obstacle and affects our vessel's maneuvers. By simple logic and reasoning it is easy to agree that it is more dangerous to be directly in front of an obstacle than behind it. Therefore, the parameter $\boldsymbol{a}$, which spans forward, is larger than $\boldsymbol{b}$ which spans backwards. The parameter $\boldsymbol{c}$ extends the penalty region to the obstacles starboard side. The elliptical penalty region makes the BC-MPC *COLREGs-aware*, not *COLREGs-compliant*, through this scheme without applying direct logic statements. The elliptical penalty region is penalizing being on the starboard side of an obstacle causing our ship to make a starboard turn in a head on situation. The head-on situation is a fairly common situation to occur at sea and the resulting action is therefore important.

With the thought of the circular region being best suited for static obstacles and the elliptical for the moving obstacles, it would be fair to suggest that making the parameters of **a**, **b** and **c** dependent on the speed of the obstacle is a good idea. However, this is not pursued further in this thesis. In [7] the tuning parameters of the BC-MPC indicates that the desired minimum distances to other vessels is in the range of tens to several hundred meters. In this thesis the margins have been reduced to be able to evaluate performance in cluttered situations.

The weighting parameters which inflict how the vessel prioritises between collision avoidance, trajectory alignment, speed deviation and course deviation has been selected by trial and error. The weight w_p is set to 1, for simplicity as the selection of the other values are dependent on the relation between the weights.

6.4 Tuning of the SB-MPC algorithm

The simulation parameters used on the SB-MPC algorithm in this thesis is presented in Table 6.3.

6.4.1 Tree generation

In the SB-MPC algorithm the tree is more easily defined, and the suggestions of 13 straight-line course trajectories for each speed alternative has been employed. A finer grid was also investigated, but the performance was not especially rewarding

Table 6.3: Selected tuning parameters for the SB-MPC algorithm

Description	Symbol	Value	Unit
Number of speed maneuver alternatives	N_U	3	
Number of course maneuver alternatives	N_χ	13	
Course offset limits		[-90,90]	°
Number of times the tree is branching	N	1	
Length of timestep	T	0.1	s
The length of a single subtrajectory	T_{end}	45	s
Maneuver time	T_U and T_χ	0	s
Horizon	T_{full}	45	s
Characterization parameter for close situations	d_{close}	200	m
Characterization parameter for immediate situations	d_{safe}	60	m
Weight of collision penalty	K_{coll}	0.5	
Weight of base cost of collision	C_base	10	
Risk parameter for time	p	0.5	
Risk parameter for distance	q	2.0	
COLREGs weight	κ	3.0	
Speed deviance weight	K_P	2.5	
Course deviance weight	K_χ	3.0	
Weight of speed change	$K_{\Delta P}$	1.0	
Weight of course change to starboard	$K_{\Delta\chi,\text{starboard}}$	0.9	
Weight of course change to port	$K_{\Delta\chi,\text{port}}$	1.2	
LOS Lookahead distance	Δ	100	m
Ahead	ϕ_{ah}	15	°
Overtaking	ϕ_{ot}	68.5	°
Head-on	ϕ_{ho}	22.5	°
Crossing	ϕ_{cr}	68.5	°

due to the cost of increased computational load. Such a grid may also cause less observable maneuvers. Hagen concludes in [14], after exploring a finer grid, that the original grid is well suited for the algorithm.

The 13 course offsets are evenly distributed in the interval $[-90°, 90°]$. As these are simply offsets from the nominal course χ_d it is in practice also impossible for the vessel to travel away from its destination. However, in very complex and dangerous situations temporarily moving away from the target, might be in the best interest of the vessel to ensure collision avoidance. Widening the course offset limitations or initializing the tree in another way might allow for this to happen, however this is not explored in this thesis.

6.4.2 Tuning of the cost function

The safety distance, d_{safe}, is increased from Hagen's suggested 40 m to 60 m. The rest of the tuning has attempted to reduce the cost of being inside this zone. This has caused the vessel to act earlier, and cause more predictable behavior. In general it seems that the minimum distance between our ownship and the obstacles correlate very strongly with the d_{safe} of the SB-MPC, while the BC-MPC seem to allow the distance to be significantly lower when passing behind another vessel. This seems to give the BC-MPC some possibilities to pursue the path following in a higher degree while at the same time, keeping the risk of collision under control.

In the SB-MPC algorithm, p and q from equation (5.5) decides the impact of the time until the risk occurs and the distance of obstacles on the risk evaluation, respectively. The time weight $w_i(t)$, which we have introduced in (4.24), plays a similar role in the BC-MPC algorithm. With $p < 1$, the effect of time is damped by its exponent. This may have a good purpose as one would like the risk to fall with time, but not too fast. However, if one lets p decline towards zero the risk over time stays constant given a fixed distance to the obstacle. Selecting $p = \frac{1}{2}$ seems like an appropriate value based on both the plots of the risk and experimental simulation with the vessel model. The tuning values of the SB-MPC algorithm are presented in Table 6.3 and 6.4.

In Figure 6.3 one can observe the risk as a function of time, with a fixed distance of 30 m. Note that the risk with the original tuning proposed by Hagen the risk is drastically larger than the limitations from the y-axis. As the time into the future

Table 6.4: Risk tuning in the SB-MPC algorithm

Description	Hagen's value	New value
q	4	1.5
p	1	0.5

t is small enough in relation to t_0 the risk will approach infinity. However, such huge values initially is not desirable as the rest of the horizon then is undermined.

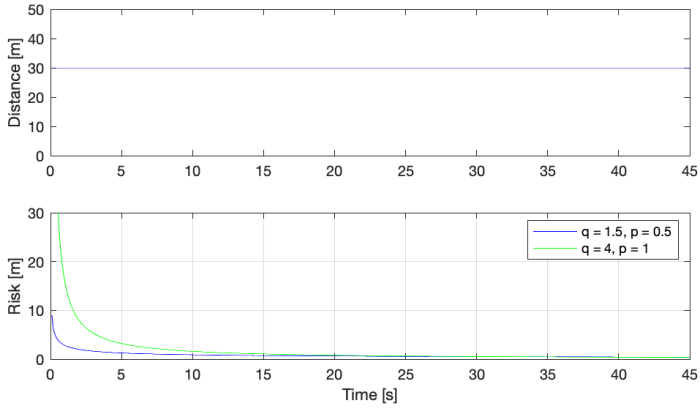
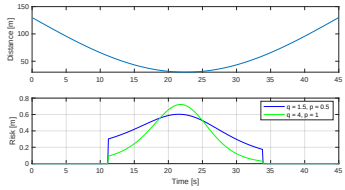
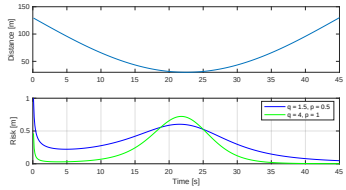


Figure 6.3: Evaluation of p and q as a function of time with constant distance $d_{0,i}^k(t)$

In Figure 6.4b the two tuning suggestions' risk is plotted against time while being subject to dynamic distance to the obstacle. This is a typical situation for the vessel to be in, and one can see that the new tuning, clearly indicates a high risk in relation to the shortest distance, while also prioritizing events that are in near future first.



(a) Risk evaluation at dynamic distance with cut off to zero when the distance is larger than d_{safe}



(b) Risk evaluation at dynamic distance without cutoff for visualization purposes

Figure 6.4: Risk evaluation with dynamic distance over time

The parameter K_χ has been increased from Hagen's 1.3 to 3, thus forcing the ownship to limit the use of course offsets. This allowed for faster convergence back to nominal course without any course offsets, while at the same time allowing for course to be the main tool for collision avoidance. $K_{\Delta P}$ and K_P, the speed tuning parameters must be selected carefully, and in relation to each other. Large values of $K_{\Delta P}$ causes the speed to be less subject to change, both from the nominal speed and back to it. Large values of K_P causes the cost of changing the speed away from the nominal speed U_d to increase. Too low values has the effect of the vessel simply using speed to resolve the situations at hand. This happens, unfortunately, at the cost of progress towards its end destination. Simulations indicate that K_P should be larger than K_χ.

The COLREGs transitional cost presented in subsection 5.3.5 is not a part of the simulation study. The grounding function g is not a part of the implementation either as it is not a topic of investigation in the thesis.

It becomes evident that the tuning is very complex and intertwined process. Changing one parameter may cause the rest of them to need adjustment as well. The tuning proposed in this thesis is therefore a suggestion that seems to work well for a wide variety of situations which we will now investigate further in Chapter 7.

Chapter 7

Simulation study

With both collision avoidance algorithms in place, the overall system with guidance law, vessel model and dynamics will now be tested, and the performance of each algorithm evaluated. The differences and challenges of both algorithms will be highlighted and discussed in relation to their functions.

7.1 Selection of simulation scenarios

We will use 8 different scenarios to evaluate the performance of the SB-MPC and the BC-MPC algorithms. The first five scenarios are single-obstacle scenarios designed to demonstrate the algorithms performance with respect to COLREGs. As Eriksen states in [7], the collision algorithms that claim to be COLREGs-*compliant*, are simply COLREGs-*aware*, meaning that their performance is not complex and customized enough for the behavior to be truly compliant.

The three complex multi-vessel scenarios' purpose is to create a sort of chaos where our ownship must actively steer and maneuver to resolve the situations in a safe way. The situations are realistic, yet challenging. There are several simultaneously active COLREGs rules where the vessel must prioritize its potential maneuvers. Situations where the obstacles fail to comply to their obligations as imposed by the rules occur. While the scenarios should be realistic, the obstacles are still assumed to travel linearly at constant speed. If they had been allowed to act unpredictably,

this would have made the scenarios even more challenging. An overview of the initial poses of the vessels in each scenario is presented in Table 7.1 and 7.2.

Table 7.1: Initial values for single obstacle scenarios

$\eta(t_0)$	$\nu(t_0)$	$\eta_{\text{Obs1}}(t_0)$	ν_{Obs1}
Scenario 1: Head on			
$\begin{bmatrix} 0\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,\text{m/s} \end{bmatrix}$	$\begin{bmatrix} 400\,\text{m} \\ 0\,\text{m} \\ 180\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ/\text{s} \end{bmatrix}$
Scenario 2: Crossing from port			
$\begin{bmatrix} 0\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ/\text{s} \end{bmatrix}$	$\begin{bmatrix} 300\,\text{m} \\ -300\,\text{m} \\ 90\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ/\text{s} \end{bmatrix}$
Scenario 3: Crossing from starboard			
$\begin{bmatrix} 0\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ/\text{s} \end{bmatrix}$	$\begin{bmatrix} 300\,\text{m} \\ 300\,\text{m} \\ 270\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ/\text{s} \end{bmatrix}$
Scenario 4: Overtaking			
$\begin{bmatrix} 0\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ/\text{s} \end{bmatrix}$	$\begin{bmatrix} 120\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 2\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ/\text{s} \end{bmatrix}$
Scenario 5: Being overtaken			
$\begin{bmatrix} 0\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ/\text{s} \end{bmatrix}$	$\begin{bmatrix} -200\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 10\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ/\text{s} \end{bmatrix}$

Table 7.2: Initial values of complex multi-vessel scenarios

$\eta(t_0)$	$\nu(t_0)$	$\eta_{\text{Obs1}}(t_0)$	ν_{Obs1}	$\eta_{\text{Obs2}}(t_0)$	ν_{Obs2}	$\eta_{\text{Obs3}}(t_0)$	ν_{Obs3}
Scenario 6							
$\begin{bmatrix} 0\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$	$\begin{bmatrix} 300\,\text{m} \\ 350\,\text{m} \\ 270\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$	$\begin{bmatrix} 200\,\text{m} \\ -250\,\text{m} \\ 90\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$	n/a	n/a
Scenario 7							
$\begin{bmatrix} 0\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$	$\begin{bmatrix} 300\,\text{m} \\ 0\,\text{m} \\ 180\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$	$\begin{bmatrix} 500\,\text{m} \\ 200\,\text{m} \\ 180\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$	$\begin{bmatrix} 600\,\text{m} \\ -20\,\text{m} \\ 180\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$
Scenario 8							
$\begin{bmatrix} 0\,\text{m} \\ 0\,\text{m} \\ 0\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$	$\begin{bmatrix} 400\,\text{m} \\ 0\,\text{m} \\ 180\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$	$\begin{bmatrix} 350\,\text{m} \\ -200\,\text{m} \\ 135\,^\circ \end{bmatrix}$	$\begin{bmatrix} 5\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$	$\begin{bmatrix} 400\,\text{m} \\ 200\,\text{m} \\ 225\,^\circ \end{bmatrix}$	$\begin{bmatrix} 3\,\text{m/s} \\ 0\,\text{m/s} \\ 0\,^\circ\text{/s} \end{bmatrix}$

For each of the 8 different scenarios presented in this chapter, the figures used to display key parameters are the same. The first figure, like Figure 7.1, contains a map over the situation area in which our ownship always starts in $\boldsymbol{p} = [0, 0]^T$. Its trajectory is drawn as a black line, while the obstacle trajectory is marked as a red line. The vehicle and obstacle starting position is drawn as a black and red circle, respectively. The other vessels i.e. the obstacles, are numerated as 1 - red, 2 - green and 3 - blue. As it is difficult to interpret where the vessels are in relation to each other at different times, the vehicle position and obstacle position at three different time instants is marked with numbers. The yellow vehicle figure and red obstacle figure are not drawn to scale.

7.2 Single obstacle scenarios

7.2.1 Scenario 1: Head on

In Scenario 1, both vessels share the duty of giving way and both shall do so by changing their course to starboard, as dictated by COLREGs rule 14. Our vessel, with the SB-MPC algorithm does so nicely with a constant course offset of approximately 45 degrees. One the vessels are well clear of each other the convergence back towards the nominal path is initiated and performed immediately. The minimum distance between the vessels is 65 m. The speed is kept at the nominal values throughout the situation.

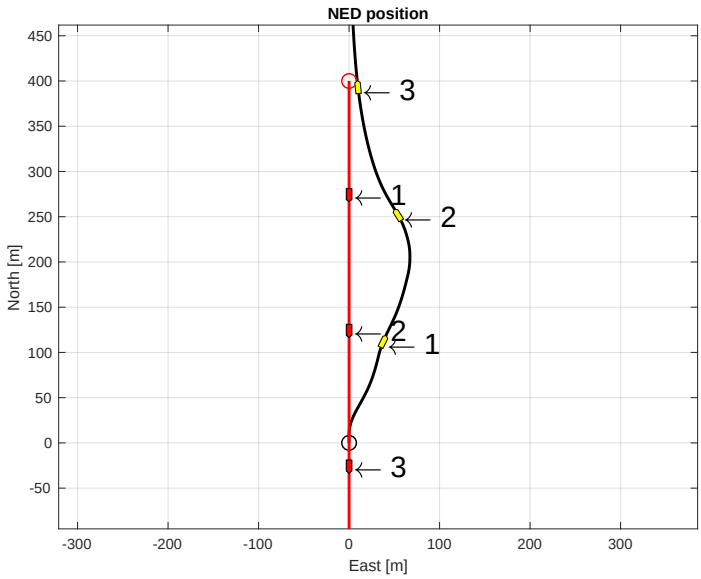


Figure 7.1: Scenario 1: Head-on with SB-MPC

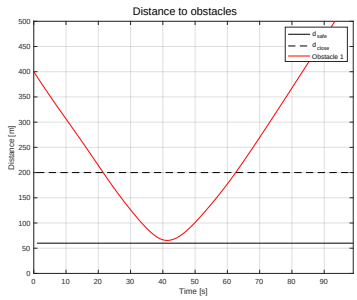
The distance plot in Figure 7.2a, displays the distances between each obstacle and our ownship against time. The threshold values d_{safe} and d_{close} is also plotted.

Each of the algorithms employs a cost function to evaluate the severeness of the

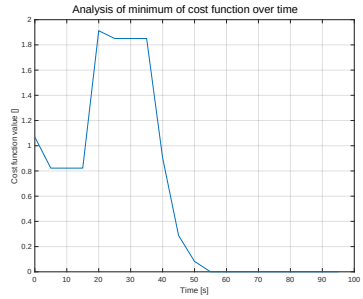
current situation. The value of the selected control input also evaluated. In SB-MPC, in Figure 7.2b, a plot of the cost of the selected alternative against time, one may evaluate the severeness of the situation the ownship finds itself in. A value of zero indicates no penalty, meaning no hazard and no selection of control inputs other than the nominal values. Typical low values in the min hazard plot indicates that there is something dangerous in the surrounding area, but the algorithm is able to avoid it by simply selecting the control scenario k which minimizes the cost function. This results in speed and course references fairly close to the nominal ones, thus achieving low or no hazard. Values that breach a threshold of approximately 5 indicate that the vessel sees no alternatives that does not bring with it any hazard, and the size of the value indicates the severeness of the situation. When it comes to the BC-MPC, which can be seen in Figure 7.4b, the same analogy applies, but the numerical values are generally higher due to the different cost function.

In Figure 7.2c and 7.2d, the course and speed are plotted. In pink, the nominal choices χ_d and U_d are presented. The blue line shows the χ_{ref} and U_{ref} which is the values presented to the low-level controllers. In green, the actual values of χ and U are presented. In SB-MPC the speed and course are assumed changed instantaneously, while in BC-MPC the selected speed and course which are used in the evaluation of the alternatives are denoted $\chi_{\text{ref,smooth}}$ and $U_{\text{ref,smooth}}$. The times of initiation of a new maneuver sequence and initial value $\chi_{\text{ref,smooth}}(t_{\text{call}})$ and $U_{\text{ref,smooth}}(t_{\text{call}})$ are marked with an red asterix. The general tendency of these red overlapping segments reveal the degree of coherency between one iteration's plan and the next as well.

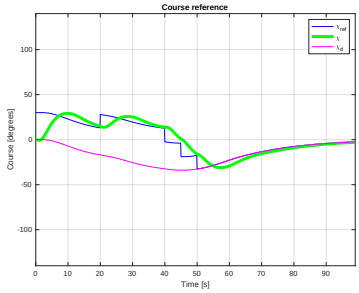
In the following sections, only the figures which play an explicitly important role or reveal an interesting feature or effect will be commented, while all of them are included to create an holistic view of the situations.



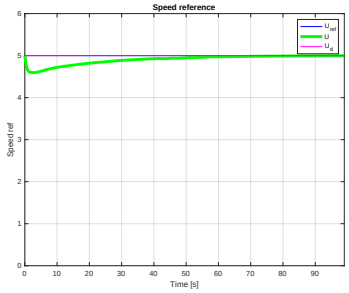
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.2: Scenario 1: Head-on with SB-MPC situational metrics

In the case with the head-on situation with the vessel controlled by the BC-MPC algorithm, the results are very similar to that of the SB-MPC. This is a good thing, as they both handle the situation in a desirable way. One can however notice that the BC-MPC initially acts slightly less, but ends up with a course of approximately 75 degrees, which is recognizably larger than that of the SB-MPC. Course trajectories from the tree of courses are selected for the first 4 calls of the BC-MPC, before, as the risk of collision disappears, the nominal alternative is selected for the rest of the simulation. This can be seen in Figure 7.4d. The minimum distance is between the vessels with the BC-MPC algorithm is 82 m. The distance is plotted against time in Figure 7.4a. The figure also contains the minor axis limits of the elliptical penalty region.

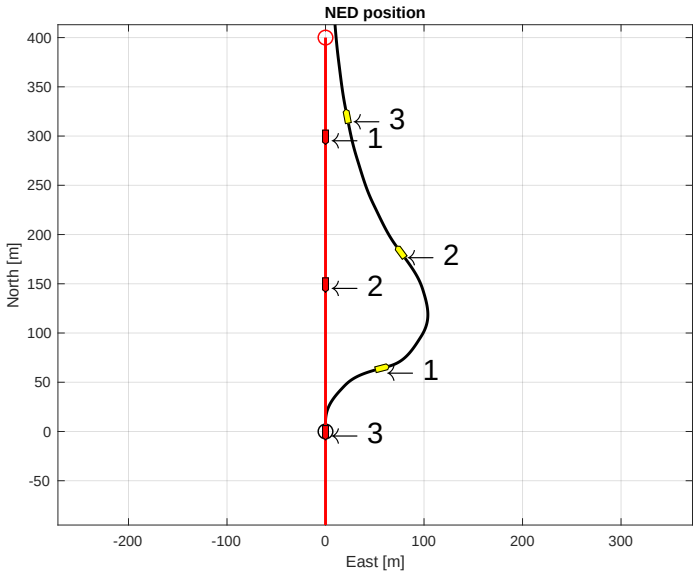
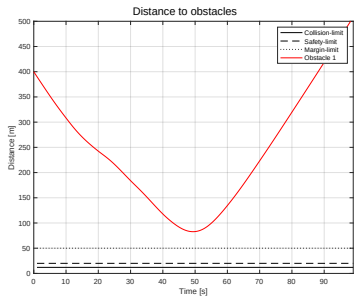
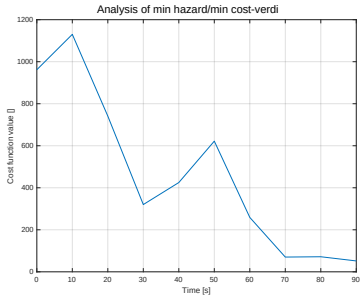


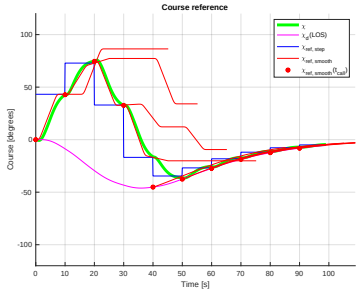
Figure 7.3: Scenario 1: Head-on with BC-MPC



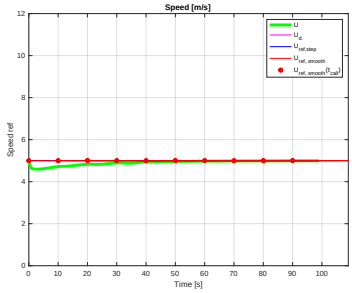
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.4: Scenario 1: Head-on with BC-MPC situational metrics

7.2.2 Scenario 2: Crossing from port

In Scenario 2, Rule 15 regarding crossing situations applies. As we are the vessel that has the other vessel entering from our port side, we have the right of way as the *stand-on* vessel. However, the *give-way* vessel does not adhere to her obligations. With the situation turning out in this manner, our ownship must do its best to reduce risk and maneuver clear of the obstacle. The SB-MPC algorithm does so by keeping the fastest speed it can choose from within the SB-MPC regime and steers off 45 degrees to starboard to prolong time to closest point of approach (TCPA) and maximize the distance at the closest point of approach (CPA) through its actions. Here it would be nice to be able to speed up above the nominal speed U_d to avoid the non-responsive obstacle. This could have been an option if values of $P_{\text{ca}} > 1$ where available for selection. The minimum distance is 66 m.

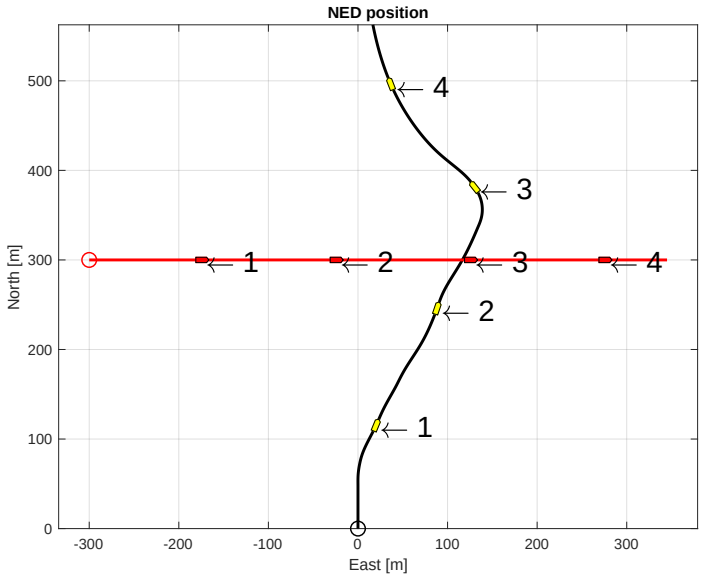
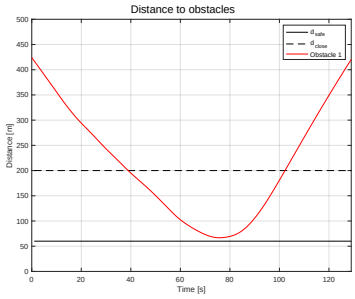
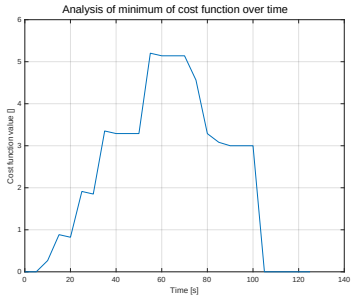


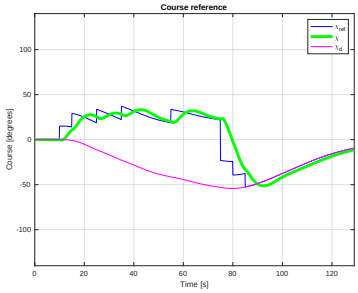
Figure 7.5: Scenario 2: Crossing from port with SB-MPC



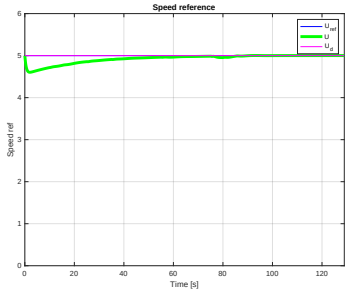
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.6: Scenario 2: Crossing from port with SB-MPC situational metrics

In Scenario 1 we observed that the elliptical penalty region helps the vessel perform maneuvers that are in line with the COLREGs. However, this region may cause our ship to perform maneuvers away from our nominal course and speed when an obstacle is crossing from port, as in scenario 2. Rule 15 clearly determines that we have the right of way in this situation. Rule 17 states that if one is in a situation where the other vessel is giving way, then you shall, as the *stand-on* vessel strive to maintain your course and speed for the purpose of predictability. However, in this situation, the *give-way* vessel does not give way, and it becomes up to us to avoid the obstacle in a safe way. The BC-MPC algorithm chooses to slightly lower the speed and steer clear of the obstacle by selecting a trajectory aft of it as can be seen in Figure 7.7. The minimum distance is 62 m. As the BC-MPC algorithm allows for increased speed, passing in front of the obstacle with increased speed may well also have been a viable action.

One may discuss whether or not the actions of the two algorithms are too early, or at an adequate point in time. This is a matter of tuning, and as both algorithms avoid the collision in a neat and predictable way, it may be evaluated to be a fair course of action.

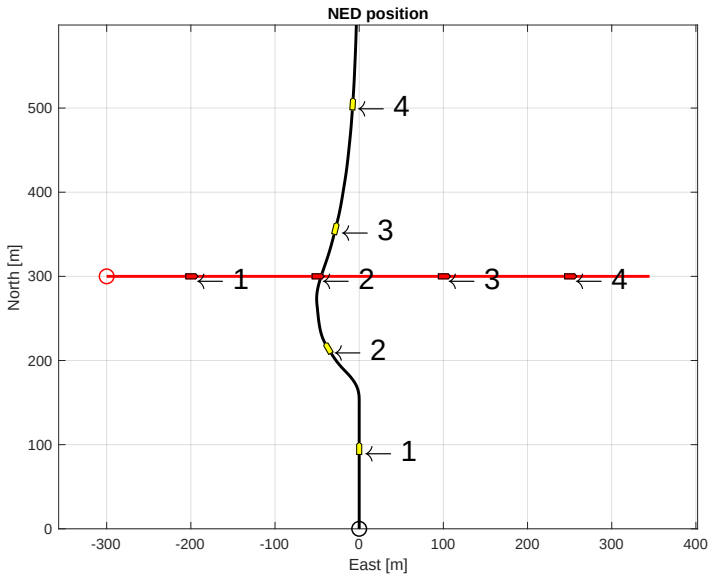
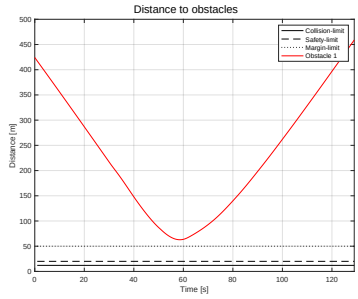
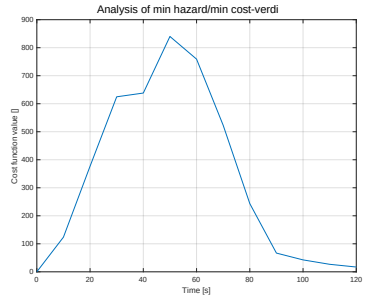


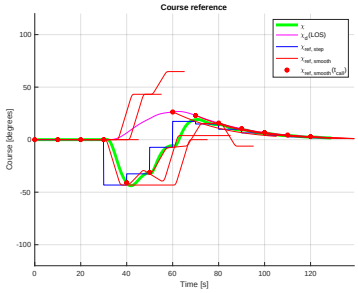
Figure 7.7: Scenario 2: Crossing from port with BC-MPC



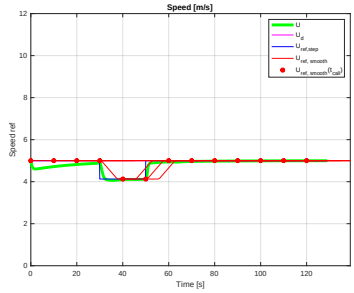
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.8: Scenario 2: Crossing from port with BC-MPC situational metrics

7.2.3 Scenario 3: Crossing from starboard

The situation in scenario 3 is not very unlike scenario 2, but although the initial positions are symmetrical identical , the actions of the algorithm are not the same. This is due to differences in penalty regions, COLREGs Rule 15 regarding crossing and tuning parameters causing different values for the different potential maneuvers.

When our ownship equipped with the SB-MPC algorithm is faced with an obstacle which is crossing from starboard, it is the give-way vessel. The way it resolves the situation can be seen in Figure 7.9. It is to first make a slight port turn, and then later a large correction to starboard to pass aft of the obstacle as COLREGs demands. The reason why the vessel acts like this is because the initial selected scenario which steers to port, avoids entering the zone of COLREGs evaluation.

Once the vessel turns out to be in that zone a couple of iterations later, it will be penalized for trying to pass in front of the obstacle. The resulting maneuver is to make a rather large course adjustment and pass behind the obstacle. The minimum distance is 60 m.

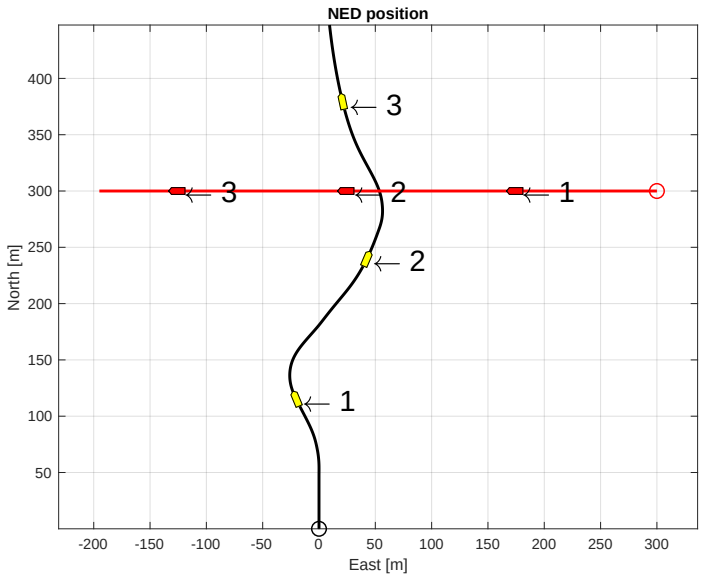
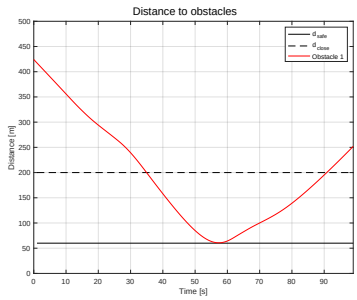
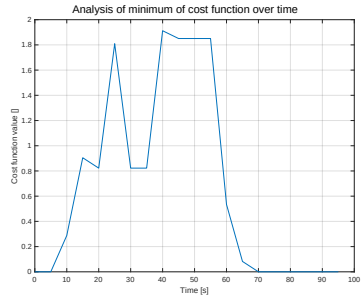


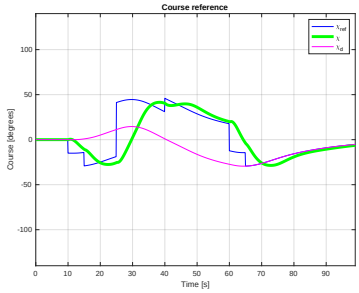
Figure 7.9: Scenario 3: Crossing from starboard with SB-MPC



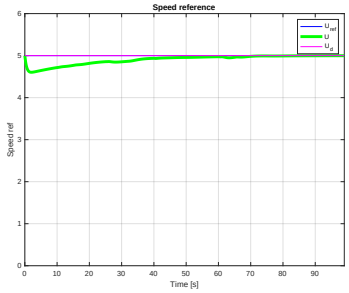
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.10: Scenario 3: Crossing from starboard with SB-MPC situational metrics

When employing the BC-MPC algorithm in Scenario 3 the vessel slows down slightly and passes aft of the obstacle. When an obstacle is crossing from starboard, rule 17 states that we are the give-way vessel. Yet due to the lack of penalty region on the port side of this vessel our ownship performs a very small maneuver to avoid the obstacle. The distance of largest deviation from the nominal path is approximately equal to that of the previous scenario, which from a safety standpoint is sufficient. However, from a COLREGs point of view it would be desirable for the algorithm to show greater will to *give way* in this scenario than in the previous. Yet, in this scenario the minimum distance is slightly less with its resulting 55 m. The performance is however adequate, due to the mid-level collision avoidance algorithms which may handle such easy single obstacle situations.

In the case that the BC-MPC algorithm goes from selecting the LOS-guidance

alternative after selecting a course and speed maneuver sequence from the tree, the reference trajectories assumes an instantaneous turn and acceleration. As this is not the case for what the real vessel may perform, the option evaluated may be artificially better than the others due to this effect. The jump in the reference may be seen in Figure 7.12c and 7.12d speed and course figure.

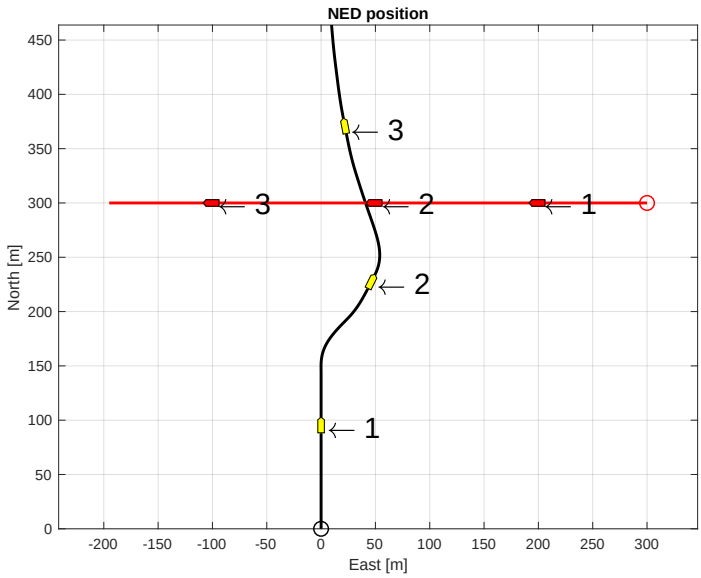
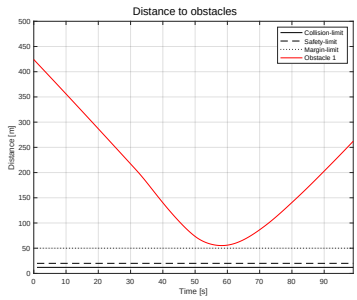
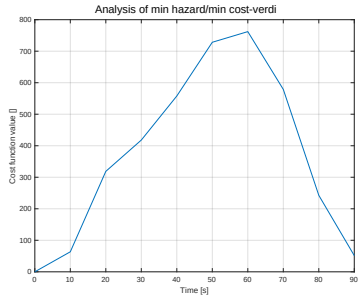


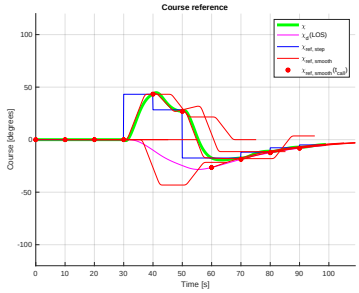
Figure 7.11: Scenario 3: Crossing from starboard with BC-MPC



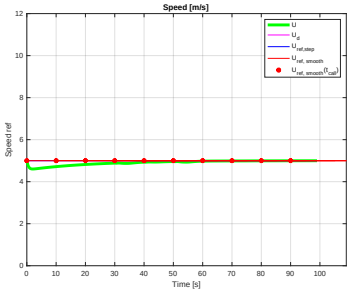
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.12: Scenario 3: Crossing from starboard with BC-MPC situational metrics

7.2.4 Scenario 4: Overtaking

In Scenario 4, when we are the overtaking vessel we also carry the obligation to keep clear of the other vessel. The SB-MPC algorithm does so by initiating a course change to starboard. The minimum distance is 66 m as the obstacle is passed. The whole situation is performed in a very smooth and predictable way. The speed is kept at the nominal value, and the course offset is fairly constant until the vessel is passed.

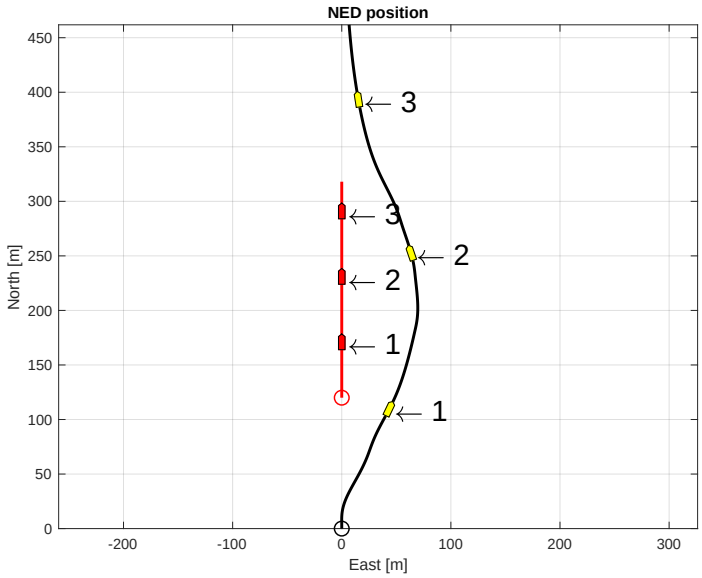
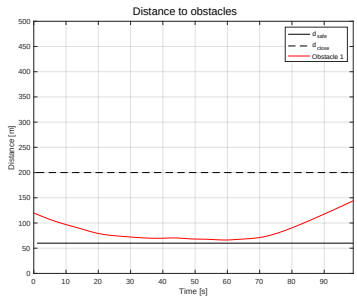
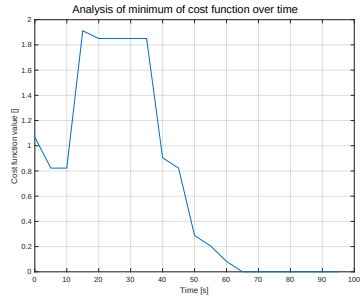


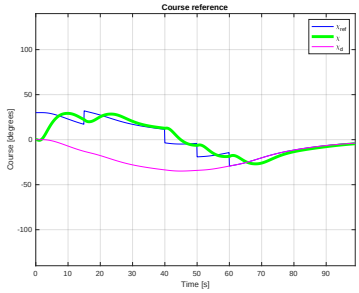
Figure 7.13: Scenario 4: Overtaking with SB-MPC



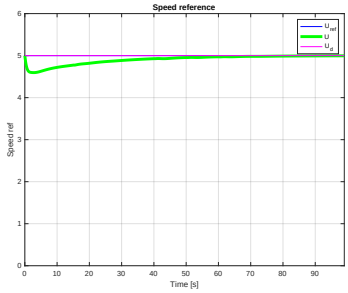
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.14: Scenario 4: Overtaking with SB-MPC situational metrics

The BC-MPC algorithm on the other hand initiates a turn to port to be able to pass the obstacle, as can be seen in Figure 7.15. The COLREGs does not dictate how such an overtaking situation must be resolved, so this choice is good as the other. The reason it does so is due to the penalty region which is spanning to the starboard side of the obstacle, causing the port side to be more attractive. One may notice that the vessel waits slightly longer before initiating the turn, but when it does, it is a very clear maneuver. The minimum distance is 48 m and the convergence towards the nominal path is slightly slower than that of the SB-MPC. This is due to the applied heuristic for the passing a future reference from the BC-MPC generated course trajectory to the controller.

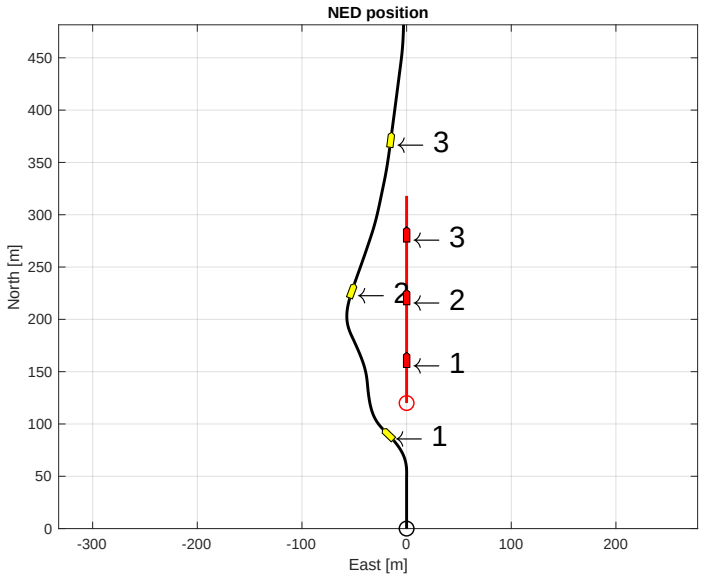
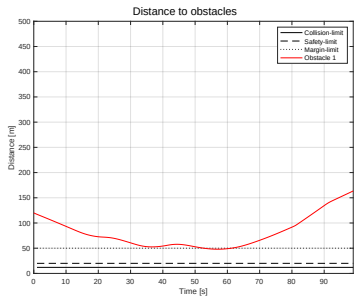
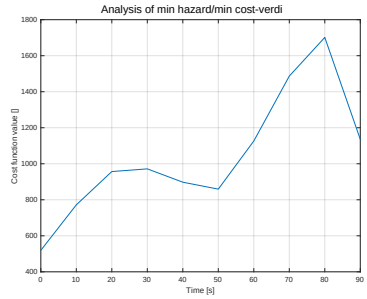


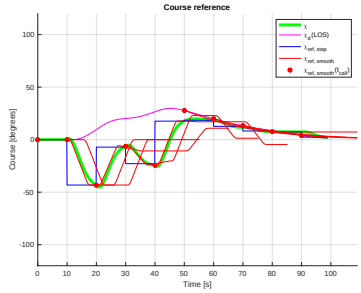
Figure 7.15: Scenario 4: Overtaking with BC-MPC



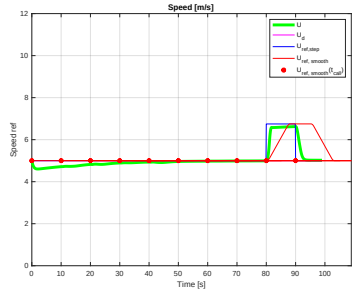
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.16: Scenario 4: Overtaking with BC-MPC situational metrics

7.2.5 Scenario 5: Being overtaken

In the case of the vessel being overtaken by another the other vessel has the responsibility of keeping clear as it is the approaching vessel which is creating the situation. In this situation, it does not do so, and it therefore becomes the job of the low-level collision avoidance algorithm to perform the necessary measures to avoid collision. Our vessel is keeping a nominal speed of 5 m/s when the obstacle is approaching from the aft at 10 m/s. The distance between them is therefore reduced rapidly and action is demanded. Our vessel chooses to maintain its speed and deviates its course to starboard. The minimum distance is 62 m as it is passed by the other vessel.

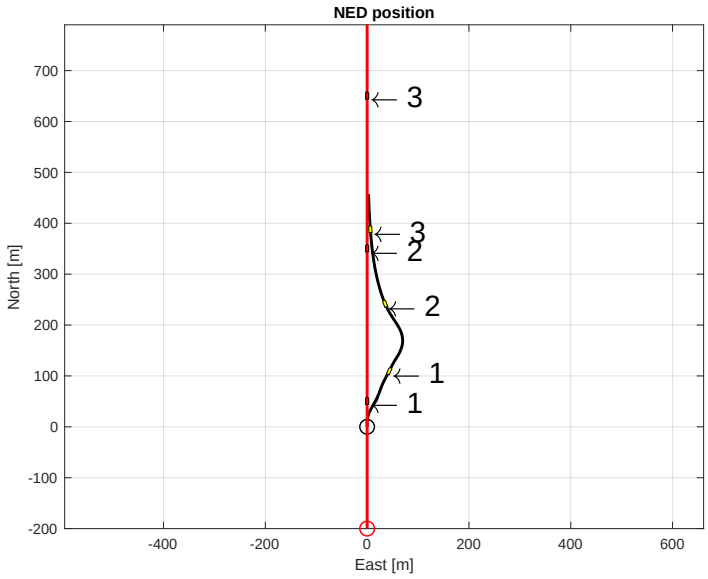
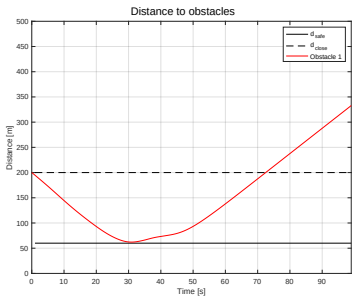
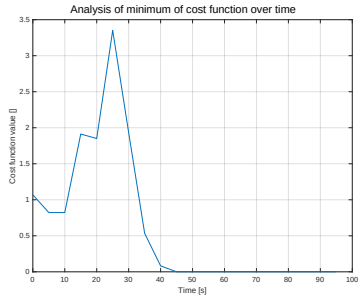


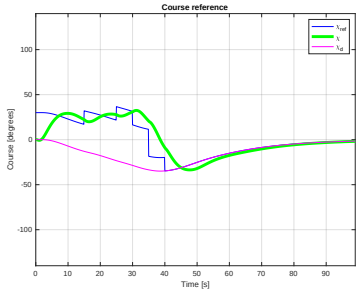
Figure 7.17: Scenario 5: Being overtaken with SB-MPC



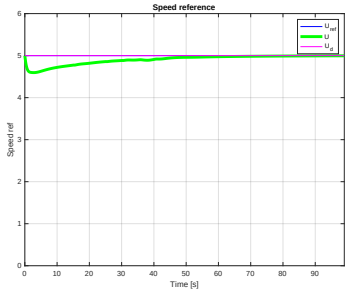
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.18: Scenario 5: Being overtaken with SB-MPC situational metrics

In this situation the vessel is forced to make large course changes from the nominal course alternative. The SB-MPC algorithm causes the vessel to maneuver in positions far from those in the selected maneuver alternative that was evaluated by the cost function. The error is typically in the range of 10 meters. This is a weakness caused by the simplifications in the SB-MPC algorithm. Yet, the algorithm handles the situations fairly well. This is due to the lacking accuracy being taken into account by having larger margins. So the resulting behavior is working in a satisfying way, but it is not operating at the same level of accuracy as the BC-MPC algorithm.

The performance of the BC-MPC algorithm can be investigated in Figure 7.19. As in the previous scenario the vessel chooses to diverge to port to avoid collision. The minimum distance between the vessels are 80 m. The course is at its largest

50 degrees off the nominal value. In Figure 7.20d while the speed is actually increased to approximately 8 m/s. This is to evade from the penalty region of the obstacle.

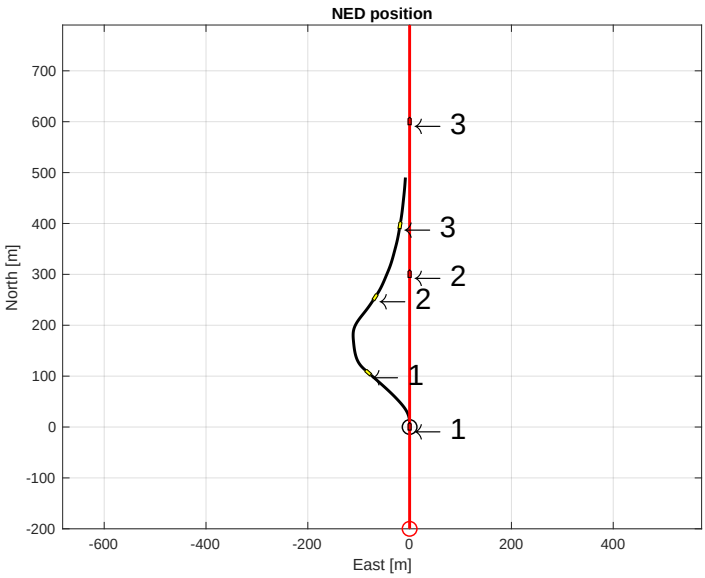
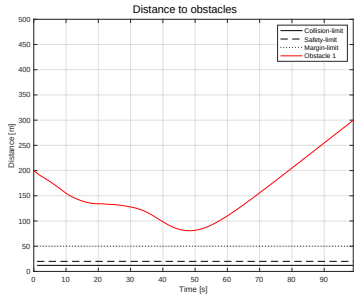
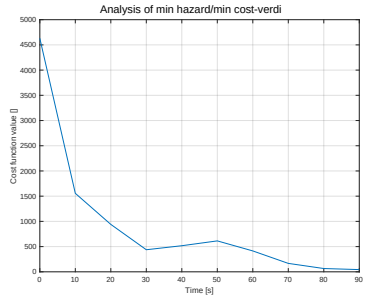


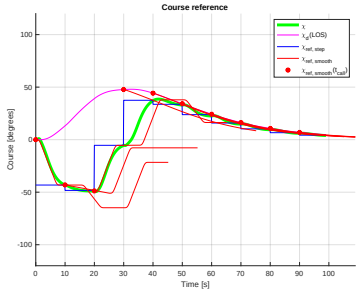
Figure 7.19: Scenario 5: Being overtaken with BC-MPC



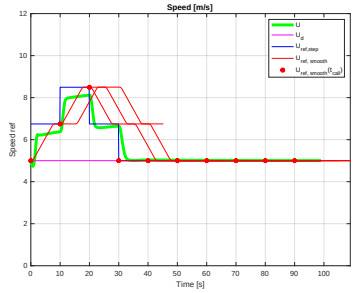
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.20: Scenario 5: Being overtaken with BC-MPC situational metrics

7.3 Multi-obstacle scenarios

7.3.1 Scenario 6: Two vessels crossing

In this scenario the vessel is faced with the challenge of two crossing vessels. The first from port (Obstacle2) which for which we are the stand-on vessel, and then one from starboard (Obstacle1) for which we are the give-way vessel. Rule 15 for crossing situation applies. The performance of the SB-MPC algorithm can be seen in Figure 7.21. The SB-MPC steers well clear of the first obstacle, passing in front at nominal speed. Our vessel then passes aft of the second obstacle heading north, before finally converging onto the nominal path. The cost function value is kept low as the vessel is able to, through its maneuvers, avoid generating any particular risk of collision. The speed is kept at the nominal value, but the course offset is

fairly large to be able to minimize the risk, as can be seen in Figure 7.22c. The minimum distances between Obstacle 1 and 2 are 61 and 86 m, respectively.

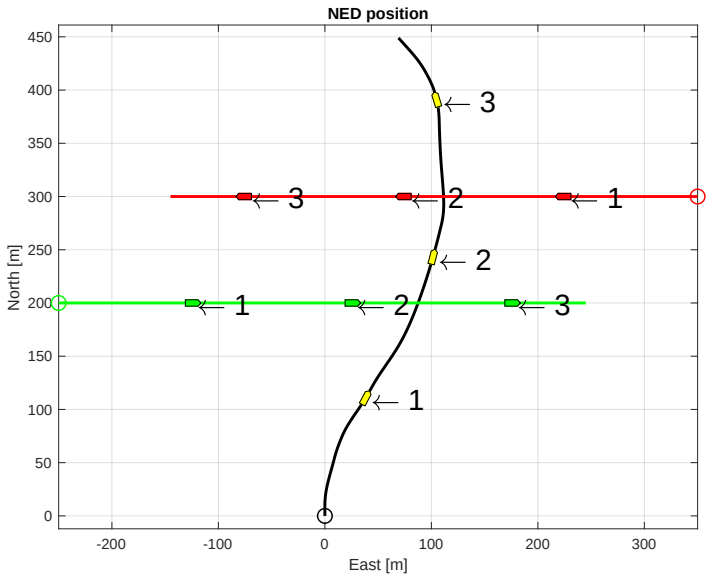
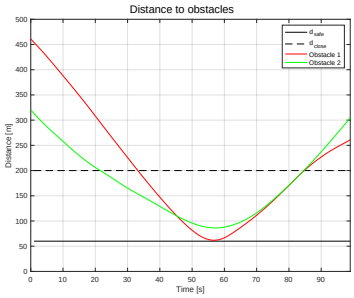
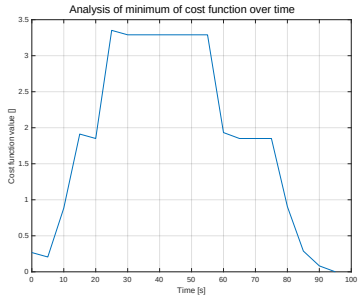


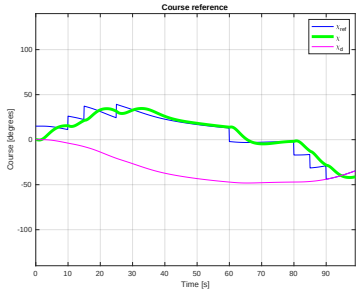
Figure 7.21: Scenario 6: Two vessels crossing with SB-MPC



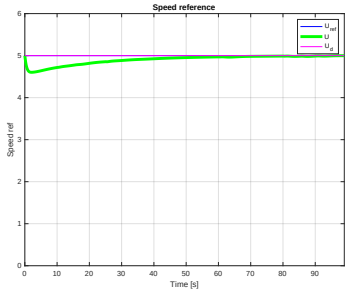
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.22: Scenario 6: Two vessels crossing with SB-MPC situational metrics

The BC-MPC algorithm resolves the situation in roughly the same way. Although one can identify some differences in Figure 7.23. The initial maneuver off from the nominal path is performed later, but with slightly larger course changes. As the vessel is passing the first obstacle it accelerates significantly up to approximately 8 m/s. This allows for the vessel to avoid the obstacles in a larger degree.

In the simulations, Obstacle 1 is the one that is the closest to our vessel. As can be seen in Figure 7.24a, the minimum distance is 88 and 139. As the minimum distances are 88m for the BC-MPC and 61 m for the SB-MPC both algorithms can be said to handle the situation well.

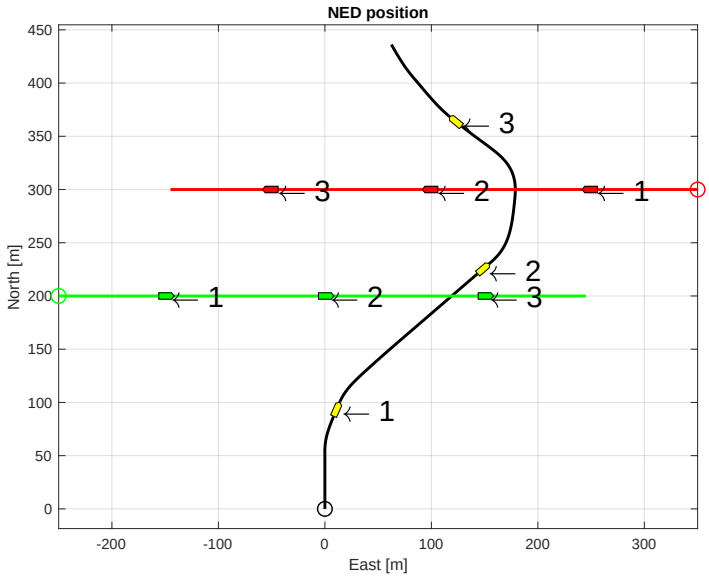
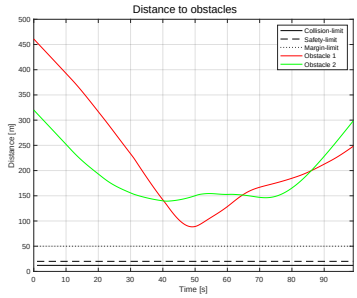
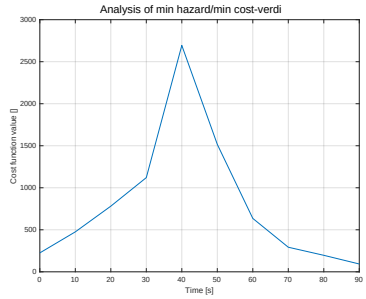


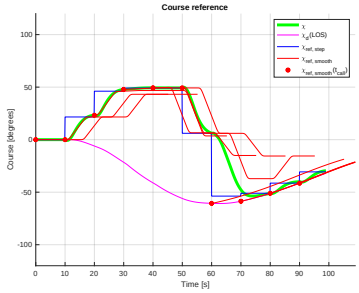
Figure 7.23: Scenario 6: Two vessels crossing with BC-MPC



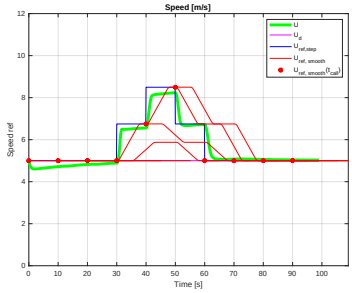
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.24: Scenario 6: Two vessels crossing with BC-MPC situational metrics

7.3.2 Scenario 7: Complex multi vessel head-on

In Scenario 7 there are three obstacles heading south, while our vessel wishes to do the opposite. As the first vessel is met head-on, SB-MPC chooses to deviate its course to starboard, as we have observed earlier. Unfortunately, this maneuver causes the vessel to approach Obstacle 2. The vessel navigates in between the two obstacles, before converging back towards the nominal path. Twice in the duration of this process does it occur that our vessel reduces its speed. This can be seen in Figure 7.26d. This aims to reduce the the distance traveled in the horizon to avoid being penalized for close encounters with the obstacles. In the first speed reduction the speed uses approximately 10 s before the actual speed is equivalent to the reference. In the second case, where the speed reference is 0 m/s from 40 to 45 s, the speed is reduced, but not nearly to the extend that is desired from the algorithms

point of view. This sort of inaccuracy could potentially be very dangerous, as it could cause the selection of maneuvers to be made on false assumptions. The minimum distances to obstacle 1, 2 and 3 is 65, 137 and 66 m, respectively.

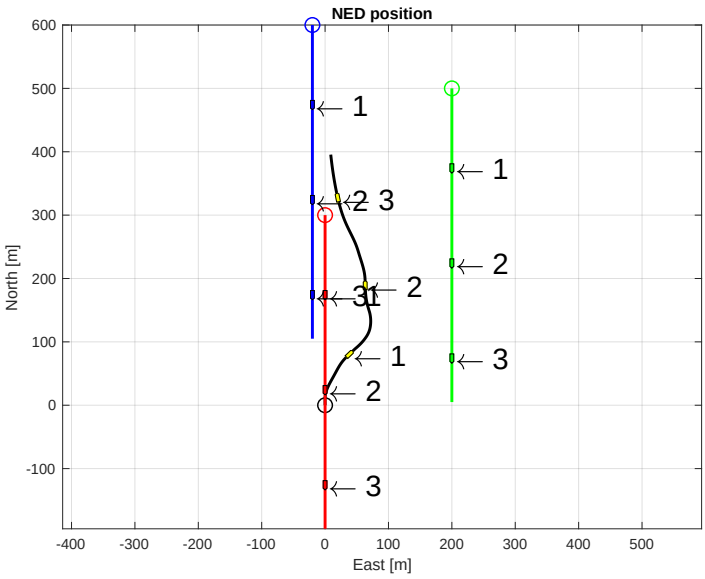
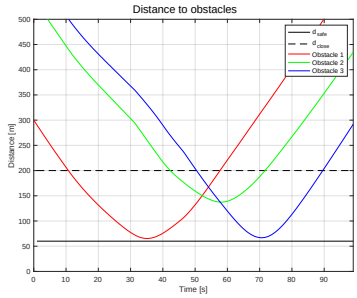
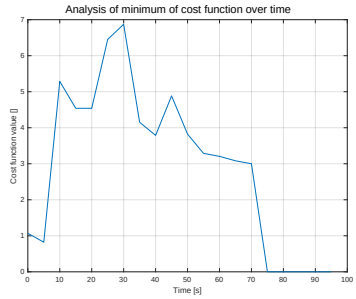


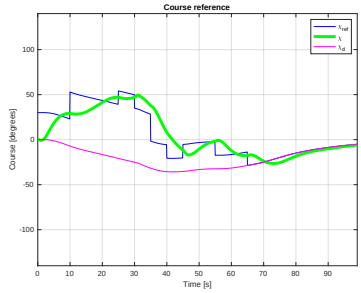
Figure 7.25: Scenario 7: Complex multi vessel head-on with SB-MPC



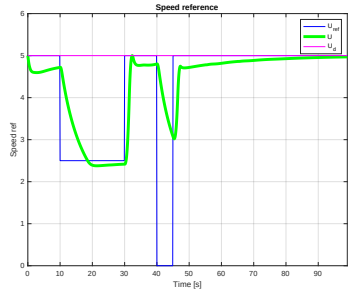
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.26: Scenario 7: Complex multi vessel head-on with SB-MPC situational metrics

In comparison to the SB-MPC the BC-MPC algorithm finds the gap between Obstacle 1 and 2 to be too narrow, and as there exists an option to deviate to port without being surrounded by obstacles, it does so. This is in violation of rule 14 regarding head on situations, but is a safe solution which allows all vessels to pass free of each other. The speed is immediately increased to be able to distance oneself from the obstacles in an efficient way. The course is selected to be approximately 140° off the nominal course for tens of seconds. As the necessary distance is created the vessel heads north again and reduces its speed to the nominal one. The minimum distances between the vessels turn out to be 153, 320 and 79 m. When the vessel decides to head port in a head on situation, it becomes evident that the elliptical penalty region causes it to have a much greater distance to the other vessel when passing.

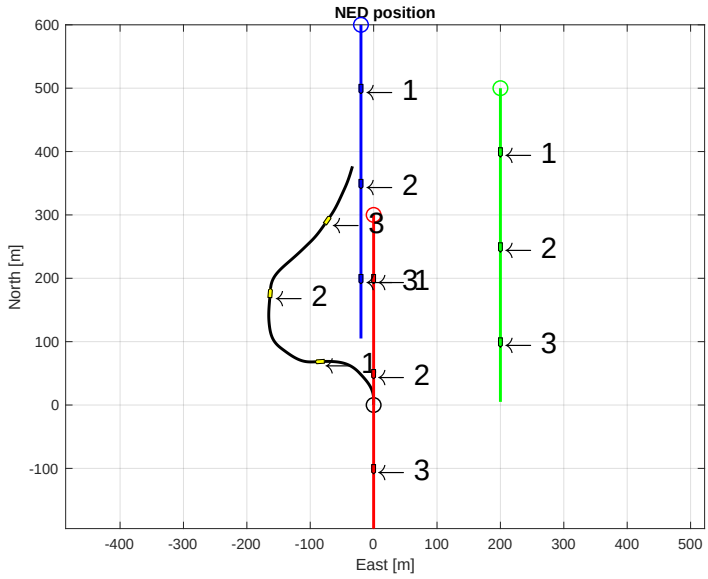
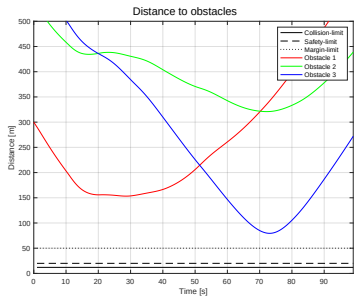
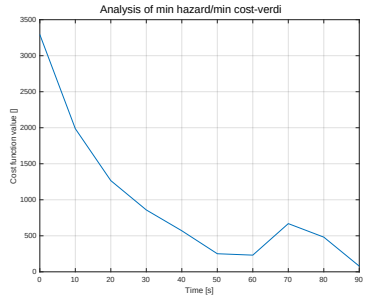


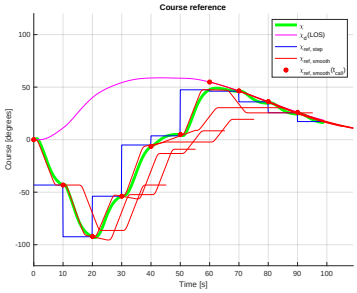
Figure 7.27: Scenario 7: Complex multi vessel head-on with BC-MPC



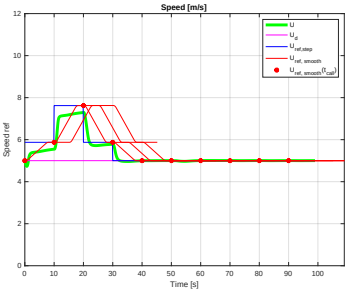
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.28: Scenario 7: Complex multi vessel head-on with BC-MPC situational metrics

7.3.3 Scenario 8: Complex multi vessel

In Scenario 8 three obstacles are approaching a point a couple of hundred meters in front of our vessel with different headings. As can be seen in Figure 7.29, the nominal path has become an unattractive one - and the SB-MPC selects an offset to starboard. The course is kept fairly stable for almost the whole horizon until the vessel is allowed to converge towards the nominal path again. Two drops in the speed reference can be observed in Figure 7.30d, and are caused due to the cost functions desire to avoid generating too much distance from the nominal path. This is a tradeoff between nominal speed and a slightly deviant course, as the course is still such that it causes the vessel to increase the vessels distance from the desired path. The minimum distance to each obstacle is 83, 115 and 62 m.

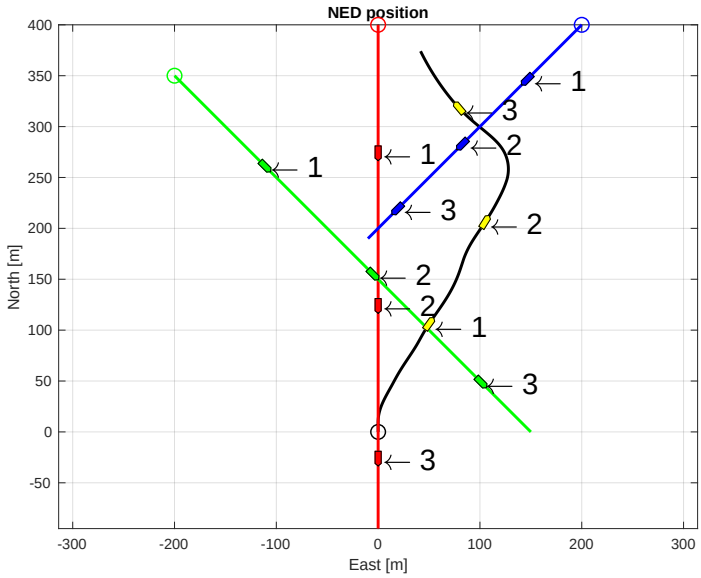
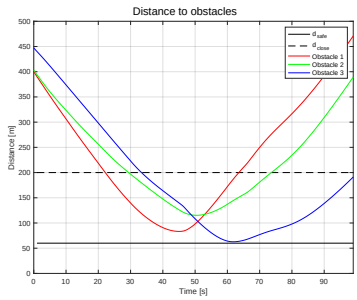
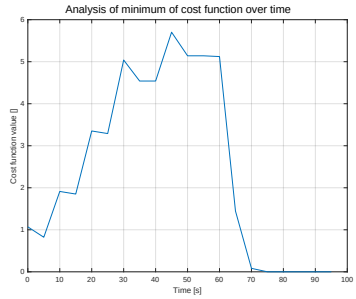


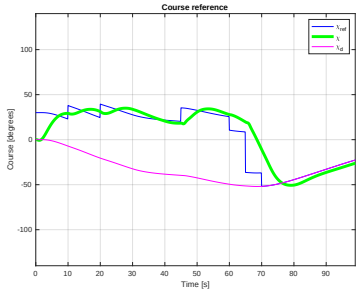
Figure 7.29: Scenario 8: Complex multi vessel with SB-MPC



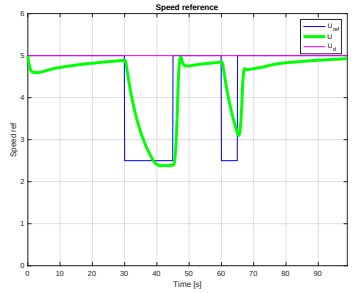
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.30: Scenario 8: Complex multi vessel with SB-MPC situational metrics

In Figure 7.31 one can see how the BC-MPC algorithm resolves the situation. It does as the SB-MPC, and diverges to starboard initially. The path taken is distancing Obstacle 1 more efficiently at first, before the course is fairly straight north for a segment. The minimum distance to each vessel is observed to be 141, 138, and 45 m. The reason our ownship passes so close aft of the last obstacle, is due to the short penalty region spanning backwards. The vessel is therefore allowed to converge with a sequence of nominal course and speed references at the same point in time as the SB-MPC, but while having spent more time at the nominal speed, resulting in overall more satisfying behavior.

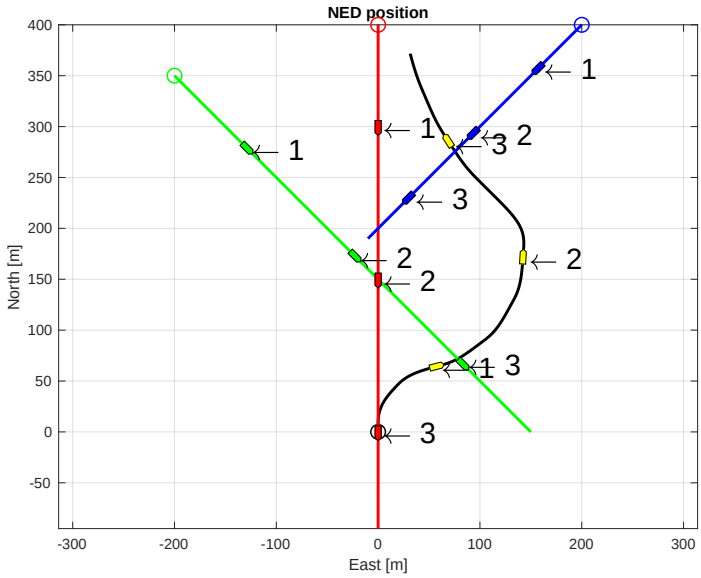
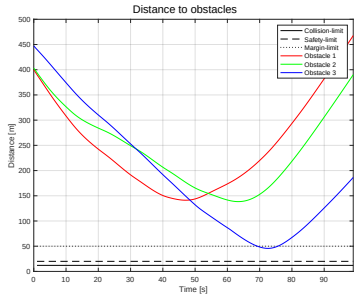
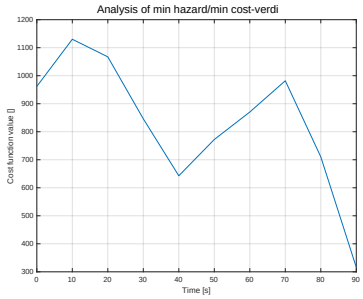


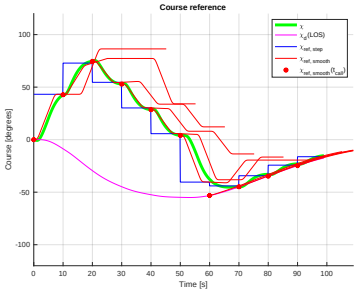
Figure 7.31: Scenario 8: Complex multi vessel with BC-MPC



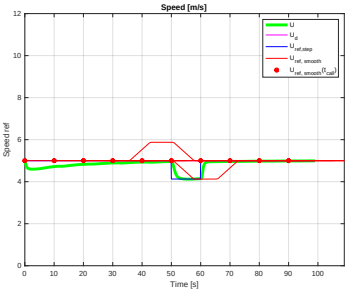
(a) Distance



(b) The cost function value of the selected scenario over time



(c) Course



(d) Speed

Figure 7.32: Scenario 8: Complex multi vessel with BC-MPC situational metrics

7.4 Discussion

The BC-MPC algorithm seems to be able to perform the different maneuvers slightly better than the SB-MPC. As they have so many equal terms of competition the increased performance is very likely to come from the development of a finer granulated tree in the BC-MPC rather than the simpler SB-MPC tree. The BC-MPC algorithm seems to generate trajectory references which relieves the situation in a slightly more effective way than the SB-MPC. Both algorithms' cost functions are intuitive to tune to cause the desired performance and they both make good choices from their tree of possibilities in most of the situations.

One way of reducing the accuracy gap between the continuous and sample based MPC is by letting the number of samples become large. If this where to approach

infinite, then almost all solutions would be represented. The exception is that the length of maneuvers in BC-MPC is defined and at the time of T_χ and T_U from the BC-MPC call, there will be a small period of constant speed and course for all alternatives. This aside, letting the number of samples grow brings with it larger computation time, causing the solution to have become "older" and not necessarily fitting to the way the situation has developed. Short computation-time is therefore important. One way of being able to solve with a fine-granulated tree and still limit computation time, is to have a coarse tree functioning as some kind of pre-investigation. Whichever trajectory in the tree that yields the lowest cost, could lay out the foundation of a fine-granulated tree in this direction and area. To further improve the performance of the current implementation of the BC-MPC algorithm the nominal LOS alternative in the maneuver tree would need a transient maneuver to be able to evaluate the process of converging towards such a path in a realistic way. This would increase the accuracy and achieve more coinciding, predicted and actual trajectories, avoiding too frequent selection of the nominal alternative.

One very clear difference between the two algorithms is their difference in evaluating risk of collision. In the SB-MPC cost function (5.3), the hazard of scenario k calculated from time t_0 is subject to two *max* operators. The first and outermost max operator selects the hazard that is associated with the most hazardous obstacle i. The second and innermost one gives the hazard at the most hazardous point in time t for each scenario with each obstacle. As a result of this H^k will not be influenced by whether the hazard is fairly large over time in the horizon, or if it occurs abruptly. This cost function will neither take into consideration whether there are several hazardous obstacles associated with scenario k or only a single one. However, in reality, being surrounded by several obstacles may limit the potential courses of action drastically thus causing a more hazardous situation. BC-MPC on the other hand, which uses a potential field method in the optimization-part (4.14) of the algorithm will simply add the different repulsive fields around obstacles thus inflicting an overall effect on the selection of the optimal trajectory. The evaluation of each point in time is summed to create an overall cost value. Maybe one would think that the behavioural differences would be larger due to this difference in trajectory selection, but they both seem to select the best possible trajectories of those available for each of them.

It is interesting to see the magnitude of the minimum distances for the different algorithms in the different situations. The safety distance d_{safe}, is the threshold for when risk is evaluated in the SB-MPC algorithm, and is in practice the desired lower limit of distance between the obstacles and the vessel. The vessel will perform evasive maneuvers in order to avoid breaching this limit. In the simulations this value has been $d_{\text{safe}} = 60m$, resulting in minimum distances just a couple of meters larger than this value in all scenarios. In BC-MPC, however, where the penalty region is dependent on the bearing between the vessels, the minimum distance is varying in much greater degree. The elliptical penalty region is defined by the vectors **a**, **b** and **c**. The resulting distances are ranging from approximately 45 to 80 m for the closest encountered vessel in the different scenarios. The advantage with this is the ability to evaluate the situation in a higher degree of accuracy and perform the best possible actions. However, if the effect of penalizing positions in front more than in aft of other vessels is exaggerated, other vessels may interpret the situation as dangerous as they are approached closely in their rear. However, in a tight and messy situation it is desirable for *our* vessel to be able to perform the safest possible actions and an accurate description of the situation is therefore practical. Another important tuning parameter is the obstacle cost gradient parameter, μ_1. It is causing an early increase in penalty as a function of distance, spanning over a much larger region than in SB-MPC. The largest region causes only marginal effects to the choice of trajectory, but this causes the BC-MPC to begin early to perform actions to avoid dangerous situations thus producing satisfying results.

As discussed in Scenario 7, the SB-MPC is clearly making decisions on false assumptions. The controlled vessel does *not* perform instantaneous turns and accelerations and the underlying assumption for tree generation does not hold. But, the algorithm performs fairly good even as this is the case. Nevertheless, it does not deliver the accuracy in computation which the BC-MPC does.

Rule 8 compliance with clear and easily observable maneuvers is allowed for if the granulation of the tree generation is not too fine. Cost function tuning that causes the vessel to make choices that are predictable is also of high importance. However, if the LOS-guidance alternative is selected, then it probably is because the vessel is in a situation of low risk of collision. In such a situation one may to a larger degree allow for a slowly changing speed and course, which may cause

poor visibility of maneuvers. In the case of SB-MPC, it will always be prone to a slowly changing χ_d from the higher-level planner. In the LOS-guidance scheme one may choose a adequately large lookahead distance Δ so that the course causes the vessel to converge slowly and predictably towards the desired path.

Chapter 8

Conclusions and future work

This chapter summarises some of the key findings in the thesis and suggests future work on the topic.

8.1 Conclusions

Through the implementation of the SB-MPC and BC-MPC algorithms on the vessel model of Odin and the simulation study performed, it becomes clear that both algorithms are truly capable low-level collision avoidance algorithms. They handle complex situations and are able to predict and plan future events, thus causing more optimal behaviour. They cause the controlled vessel to make clear and predictable maneuvers in line with COLREGs rule 8, which reduce the risk of collision. The algorithms are both COLREGs-aware, but not entirely COLREGs-compliant. This is due to the lack of COLREGs specific logic which would have to handle very complex situations. This is difficult due to the vague definitions of the COLREGs and the room for interpretations of situations. For a low-level COLAV algorithm this should not be the focus either, as its purpose is to handle emergency situations where obstacles are either detected late, behaving dangerously or the middle-layer fails to produce a valid solution to the situation.

Comparing the two algorithms, the BC-MPC algorithm is a more complex algorithm to implement, but it is able to evaluate trajectories which are of higher complexity and coincide better with the trajectories actually performed by the vessel. This proves to be a valuable property when the environment becomes cluttered, and margins are reduced. The search space of trajectories is also more refined as it allows for sequences of maneuvers in comparison to SB-MPC's single choice. This being said, SB-MPC is able to apply logic regarding its trajectories in a simpler way which allows for attractive logical COLREGs evaluation functions in the cost function, while BC-MPC is employing the repulsive fields for the same function.

8.2 Future Work

For these algorithms to become more widely known and used a systematic tuning guide for quick and easy implementation would have to be presented. This should govern different vessel types, propulsion systems and mission types for the vessels. Making more of the parameters dependent on situational parametres will give the cost functions more knowledge to be able to make better choices. This could for example be penalty regions which are dependent on the speed of the vessel and the obstacles.

Both implementations would probably gain increased accuracy in navigation through the implementation of a feedforward controller for course and speed. The BC-MPC algorithm in particular needs such a controller to truly benefit from its precisely generated trajectories. However, in the current implementation, the trajectory-generation parameters have been tuned through empirical data so that the error in the actual and projected trajectories are minimized, and the overall performance is satisfying.

Further simulations and on-water testing and verification should also be a part of the future work with the algorithms. As a part of the verification simulations one should strive to give the algorithm as equal terms of operation as in real life as possible. One of the differences one should account for is that of measurement and estimation noise. This could be generated in simulations and help ensuring a swift transition to real vessels. Although both algorithms have been tested in practice previously, both would benefit from stress testing in multi obstacle

cluttered environments where accuracy in trajectory tracking and planning abilities are explored. The integration with the vessel and its dynamics which may differ from that of the model may reveal new challenges. In practice scene analysis with obstacle detection and tracking may be noisy, thus reducing performance. Therefore, several heuristics may need to be applied for implementation and sensor integration in order to achieve great performance.

Bibliography

[1] T. Fagerlund, "Motion Planning and Collision Avoidance for USVs, TTK4551: Specialization project, 12.2019.".

[2] E. Ruud, "USV for future maritime mine counter measures: A software description," 2019.

[3] "Norges nye ubemannede minejegere," *FFI - VITEN*, vol. 1.2019, p. 32–35, 2019.

[4] M. Nakjem, R. Fardal, J. Sandrib, and J. Bjørnstad, "Unmanned surface vehicles (USV) in the future norwegian unmanned maritime mine counter measures (MMCM) concept," 2019.

[5] "Automatic identification systems (AIS)" "url=http://www.imo.org/en/ourwork/safety/navigation/pages/ais.aspx," 2019.

[6] "Marine Traffic" url = "marinetraffic.com," 2019.

[7] B. H. Eriksen, M. Breivik, E. F. Wilthil, A. L. Flåten, and E. F. Brekke, "The branching-course model predictive control algorithm for maritime collision avoidance," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1222–1249, 2019.

[8] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," pp. 49 – 60, 11 1987.

[9] O. Khatib, "The potential field approach and operational space formulation in robot control," in *Adaptive and Learning Systems*, pp. 367–377, Springer, 1986.

[10] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pp. 1398–1404, IEEE, 1991.

[11] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, pp. 760–, 07 1998.

[12] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335 – 348, 1989.

[13] B. Foss and T. A. N. Heirung, *Merging Optimization and Control.* 03 2016.

[14] I. B. Hagen, "Collision avoidance for ASVs using model predictive control," Master's thesis, NTNU, 2018.

[15] T. Johansen, T. Perez, and A. Cristofaro, "Ship collision avoidance and COLREGs compliance using simulation-based control behavior selection with predictive hazard assessment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 1–16, 05 2016.

[16] I. B. Hagen, D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, "MPC-based collision avoidance strategy for existing marine vessel guidance systems," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7618–7623, IEEE, 2018.

[17] Y. Lu, X. Zhucun, G.-S. Xia, and L. Zhang, "A survey on vision-based uav navigation," *Geo-spatial Information Science*, pp. 1–12, 01 2018.

[18] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[19] S. LaValle and J. Kuffner, "Randomized kinodynamic planning.," *I. J. Robotic Res.*, vol. 20, pp. 378–400, 01 2001.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotic Research - IJRR*, vol. 30, pp. 846–894, 06 2011.

[21] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," pp. 1478–1483, 06 2011.

[22] A. Brooks, T. Kaupp, and A. Makarenko, "Randomised MPC-based motion-planning for mobile robot obstacle avoidance," pp. 3962 – 3967, 06 2009.

[23] B.-O. Eriksen and M. Breivik, "MPC-based mid-level collision avoidance for ASVs using nonlinear programming," 08 2017.

[24] B.-O. H. Eriksen, G. Bitar, M. Breivik, and A. M. Lekkas, "Hybrid collision avoidance for ASVs compliant with colregs rules 8 and 13-17," *arXiv preprint arXiv:1907.00198*, 2019.

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

[26] R. Beard and T. McLain, *Small unmanned aircraft - Theory and practice*. Princeton University Press, 2010.

[27] D. K. M. Kufoalor, T. A. Johansen, E. F. Brekke, A. Hepsø, and K. Trnka, "Autonomous maritime collision avoidance: Field verification of autonomous surface vehicle behavior in challenging scenarios," *Journal of Field Robotics*, vol. 37, no. 3, pp. 387–403, 2019.

[28] T. I. Fossen and T. Perez, "Marine systems simulator (MSS)." `https://github.com/cybergalactic/MSS`, 2004.

Torstein Fagerlund

Sample Based Model Predictive Control for Path Following and Collision Avoidance of Unmanned Surface Vehicles

NTNU
Norwegian University of Science and Technology

FFI Forsvarets forskningsinstitutt
Norwegian Defence Research Establishment