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

Herman Berget

An Area-Time Trajectory Planning Approach to Collision Avoidance for Confined-Water Vessels

Master's thesis in Cybernetics and Robotics
Supervisor: Morten Breivik
Co-supervisor: Emil Hjelseth Thyri and Bjørn-Olav Holtung Eriksen
June 2021

Herman Berget

An Area-Time Trajectory Planning Approach to Collision Avoidance for Confined-Water Vessels

Master’s thesis in Cybernetics and Robotics
Supervisor: Morten Breivik
Co-supervisor: Emil Hjelseth Thyri and Bjørn-Olav Holtung Eriksen
June 2021

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

Abstract

This master's thesis presents a trajectory planning approach to Collision Avoidance (COLAV) for an Autonomous Surface Vehicle (ASV) operating in confined space and in the presence of other vessels. The proposed approach is based on representing all static and dynamic features of the operational area in a three dimensional space, spanned by the area and time. By doing this, the task of planning a collision free trajectory is reduced to a path-planning problem in three dimensions. A graph which represents a set of collision free trajectories that are feasible for the vessel is built. Each edge of the graph is then evaluated according to a cost function and the lowest cost trajectory is chosen. The cost function considers several aspects of the surroundings, in addition to a cost related to how compliant that edge is with respect to the The International Regulations for Preventing Collisions at Sea (COLREG). By doing this the algorithm attempts to create a safe trajectory from the start position to the destination.

Through simulations the proposed method is evaluated and compared to two other methods for COLAV, namely Velocity Obstacle (VO) and Single Path Velocity Planner (SP-VP). It is seen that the proposed method is able to create a safe and intuitive path in many of the situations, and improves on the compared methods in terms of safety at the cost of increased complexity. In some situations, notably where the assumptions about the behavior of the obstacles does not hold, the performance of the method is suboptimal. Changing environments can lead to frequent replanning where the trajectory may change in ways other seafarers do not expect.

Sammendrag

Denne masteroppgaven presenterer en baneplannleggingsmetode for Collision Avoidance (COLAV) for en Autonomous Surface Vehicle (ASV) i trange farvann og med andre fartøy i nærheten. Metoden baserer seg på å representere arbeidsområdet og alle relevante objekter i et tredimensjonalt rom som spennes av arbeidsområdet og tid. Ved å gjøre dette blir COLAV-problemet redusert til et baneplanleggingsproblem i en ekstra dimensjon. En graf som representerer et sett med kollisjonsfrie baner som er følgbare gitt karakteristikken til fartøyet blir laget. Hver kant i grafen blir evaluert ved hjelp av en kostnadfunksjon og banen med den laveste kostnaden blir valgt. Kostnadsfunksjon tar hensyn til flere aspekter av omgivelsene og prøver å evaluere hvor kompatibel med The International Regulations for Preventing Collisions at Sea (COLREG) en bane er. Ved å gjøre dette forsøker algoritmen å lage en trygg bane fra startposisjonen til målposisjonen.

Gjennom simuleringer evalueres den foreslåtte metoden og den sammenlignes med to andre metoder, Velocity Obstacle (VO) og Single Path Velocity Planner (SP-VP). Simuleringene viser at den foreslåtte metoden klarer å generere trygge og intuitive baner i mange av tilfellene. Den foreslåtte metoden gjør det bedre enn metodene det sammenlignes med med tanke på trygghet på bekostning av økt kompleksitet. I noen situasjoner, spesielt der antagelsene om oppførselen til hindringene ikke gjelder, yter metoden suboptimalt. Endringer i omgivelsene kan føre til hyppig replanlegging hvor banen kan endres på måter andre ikke forventer.

Preface

This thesis marks the completion of my Master's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU).

In this thesis I had the opportunity to continue my work from the specialization project and improve on the method presented there. It has been interesting and I am happy with the improvements that have been made.

I would like to thank my supervisors for feedback, suggestions and discussions throughout the semester. The proposed method and this thesis are both much better thanks to their help.

Herman Berget
Trondheim, June 7, 2021

Contents

List of Figures

List of Tables

Acronyms

AIS Automatic Identification System. 14

ASV Autonomous Surface Vehicle. iii, v, 1, 3, 14, 15, 30, 33, 36, 47

AT Area-Time trajectory planning method. 4, 30, 31, 33, 34, 36, 39, 41, 42, 45–48

COLAV Collision Avoidance. iii, v, 1, 11, 47

COLREG The International Regulations for Preventing Collisions at Sea. iii, v, 1, 2, 4–8, 22–25, 30, 33, 36, 43, 46–48

DP Dynamic Positioning. 30

MP-VP Multiple Path Velocity Planner. 3

OS Ownship. 7, 8, 18, 23, 29–40, 43–47

SP-VP Single Path Velocity Planner. iii, v, 1–3, 29, 30, 33, 36, 37, 45–47

TS Target ship. 7, 8, 23, 29–40, 43, 45, 46

VO Velocity Obstacle. iii, v, 11, 29–31, 33, 34, 36, 38, 45, 47

Chapter 1

Introduction

1.1 Motivation

The last decades have seen a large growth in urbanization. This growth comes with a set of problems, and among them is transportation. Transportation should be efficient, economical and friendly to both the local and the global environment. Urban centers are often located near waterways. These are often underutilized and represent an opportunity to aid in the transportation problem. Technology related to autonomous systems has seen large improvements in recent years. Autonomous systems can be more efficient and economical than human controlled systems.

Autonomous ferries thus become a natural part of the solution to the transportation problem. Autonomous ferries can be flexible and fill different needs in an urban area. They do not require substantial infrastructure in the same way as bridges or tunnels do.

A necessity for an autonomous ferry is a Collision Avoidance (COLAV) system. The ferry must be able to sense its surroundings and act accordingly to avoid obstacles and reach the desired destination in an efficient way. If an autonomous ferry is to fit into an already existing trafficked area it must follow the rules of the sea and behave as other seafarers expect. In practice this means that it has to follow The International Regulations for Preventing Collisions at Sea (COLREG)[International Maritime Organization (1972)].

1.2 Previous Work

1.2.1 Single Path Velocity Planner

The work in this thesis builds on the work presented in [Thyri et al. (2020)]. There, a path-velocity decomposition approach is used. The method plans a velocity profile for the Autonomous Surface Vehicle (ASV) along a predefined waypoint mission from the start position to the desired goal position.

All obstacles are projected into a path-time space. The path-time space is spanned by the distance along the path on one axis and time on the other axis. In this way, the dynamic obstacles are transformed into static obstacles in a different space. Then, a

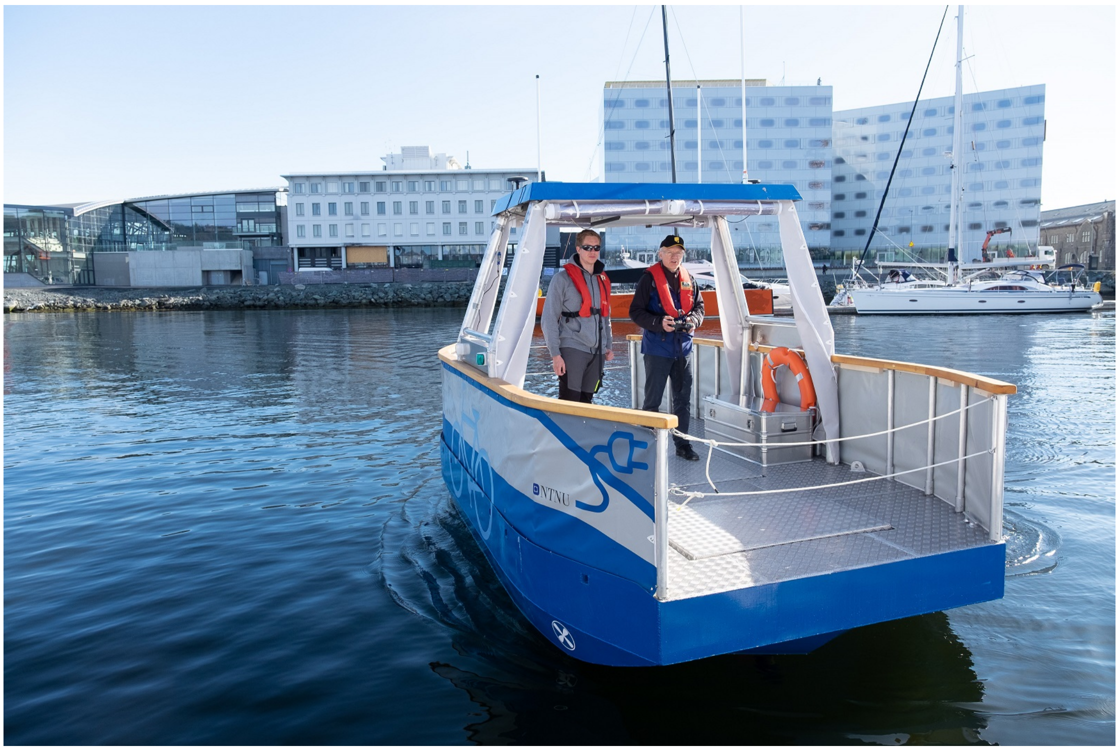


Figure 1.1: Photo of the “milliAmpere” test vehicle. Photo: Kai Dragland, taken from https://www.ntnu.edu/autoferry

graph is built in path-time space. The graph represents different velocity profiles along the path. An example graph from the article is shown in Figure 1.2.

A cost is attached to each to each node in the graph. The cost is calculated as a combination of the distance to the obstacles and the time taken to traverse the corresponding edge of the graph. From this, an optimal velocity profile along the path could be found.

This approach has a nice simplicity, but limits the possible movements of the vessel quite a bit. For example, in a head on situation where another vessel is travelling along the same path as the controlled vessel, the algorithm breaks down as it is unable to find a trajectory that traverses the predefined path without collision. Due to the predefined path, the vessel is unable to perform course changing maneuvers, and is confined to only velocity changes either forwards or backwards along the path. Such maneuvers are often undesirable or do not comply well with expected vessel behavior, which can result in confusion. Additionally, such behavior is not compliant with COLREG, where readily apparent course change maneuvers are the preferred way of maneuvering to avoid collision.

The Single Path Velocity Planner (SP-VP) algorithm in [Thyri et al. (2020)] shines when the vessel is travelling perpendicular to the traffic. In this case, there is nothing blocking the predefined path and the algorithm provides a simple and efficient method for finding a transit trajectory.

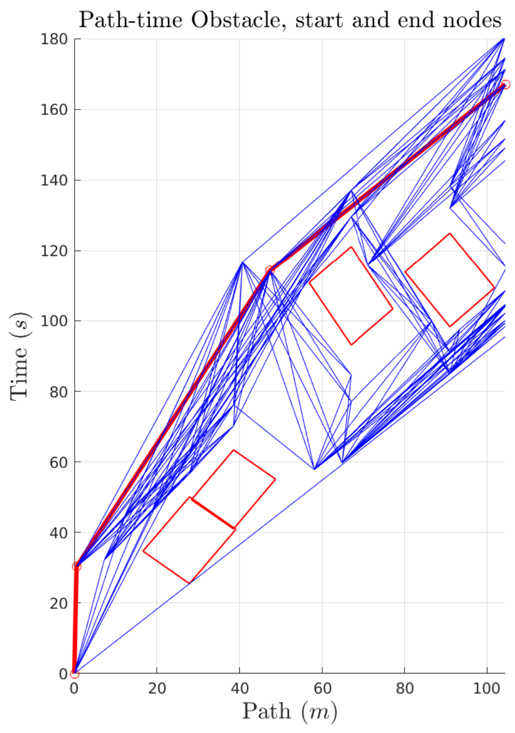


Figure 1.2: Graph created and the chosen path in path-time using the SP-VP method from [Thyri et al. (2020)]. Courtesy of [Thyri et al. (2020)]

1.2.2 Multiple Path Velocity Planner

To combat the limitations of SP-VP, [Thyri et al. (2020)] also present a modified version, Multiple Path Velocity Planner (MP-VP), where the vessel can switch between multiple predefined paths. This improves the range of motions the vessel can make, but the algorithm still suffers from most of the weaknesses of the single path algorithm.

1.3 Problem Description

The problem at hand is trajectory planning for ASV operations in confined waters. This work expands on the SP-VP algorithm [Thyri et al. (2020)] and the work done by the author in the specialization project performed during the fall of 2020.

In more detail, it concerns developing an algorithm for trajectory planning for an ASV operating in a confined working area populated by other moving vessels, where the algorithm plans a trajectory from the current position of the ASV to a desired destination. The resulting algorithm should produce trajectories that:

- Are collision free, regarding both static and dynamic obstacles.
- Ensure that the ASV keeps within the assigned operating area.

- Are feasible given the vessel characteristics. For example, the trajectory should not have velocities larger than what the vessel is constructed to be operated at.
- Adhere to the relevant protocol. For maritime operations it is the The International Regulations for Preventing Collisions at Sea (COLREG) and any local protocols that may dictate speed regulations, etc.

1.4 Contribution

- A method for building a graph that describes feasible, collision-free paths from the start position to the goal position. This method is called Area-Time trajectory planning method (AT).
- A cost function for choosing the best path from the built graph
- Validation of the proposed method through simulations with a model of the autonomous ferry prototype “milliAmpere”.

1.5 Outline

The rest of this thesis will consider the problem described in Section 1.3. Chapter 2 gives an introduction to background theory that is useful for understanding the problem and the presented solution. Chapter 3 presents a method for solving the problem. Chapter 4 presents simulation results, along with a discussion of the results. Finally, Chapter 5 concludes the thesis.

Chapter 2

Background Theory

2.1 The International Regulations for Preventing Collisions at Sea

The International Regulations for Preventing Collisions at Sea
[International Maritime Organization (1972)] are, as the name suggests, rules for how vessels should maneuver to not collide with each other.

COLREG contains 41 rules, divided into the following six parts:

Part A - General Covers definitions, responsibilities, and for whom the rules apply.

Part B - Steering and Sailing Divided into three sections:

- **Section I** Covers general rules that apply in any condition of visibility.
- **Section II** Covers rules for vessels in sight of one another.
- **Section III** Covers rules for vessels in restricted visibility.

Part C - Lights and Shapes Covers the requirements for light for different types of vessels.

Part D - Sound and Light signals Covers different sound and light signals and in which situations they should be used.

Part E - Exemptions Ships built before 1972 are exempted from some sound and light requirements.

Part F - Verification of compliance Rules for verifying compliance of the parties of the convention.

In addition, there are four annexes with technical details for sound and lights.

For the work presented in this thesis, the following subset of rules is particularly relevant:

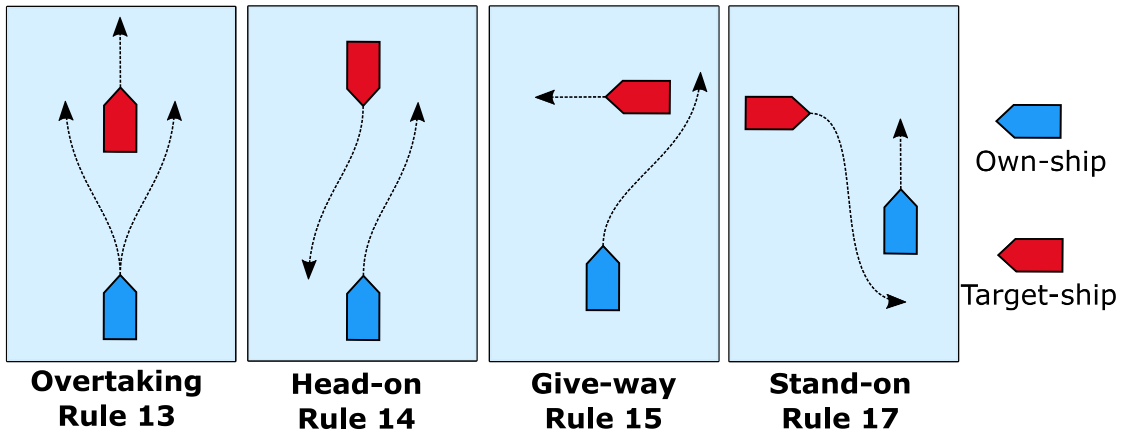


Figure 2.1: Different situations in COLREG rules 13-15 and 17. Courtesy of Emil Thyri.

Rule 6: Safe speed All vessels must keep a speed where it, given the conditions, can take proper actions to avoid collisions.

Rule 8: Early action When taking action to avoid collision, it should be taken early and be clear to other vessels.

Rule 8: Prefer course alterations If there is sufficient sea-room, altering of the course is the preferred maneuver, as it is more apparent to other vessels. If this is not enough, the vessel should also lower its speed.

These rules are general and apply in all situations. They also outline what seafarers can expect from other seafarers. Unexpected behavior may lead to dangerous situations. By acting in a predictable manner, safety can be greatly improved.

In addition to these general rules, there is a set of situations with specific rules on how a vessel shall react when meeting another vessel. These situations are:

Rule 13: Overtaking A vessel comes from behind another and wants to pass. The overtaking vessel shall keep out of the way from the other.

Rule 14: Head-on Two vessels move towards each other with opposite courses. Each vessel shall pass on the port side of each other.

Rule 15: Crossing The vessel which has the other on the starboard side shall give way. The other vessel shall stand on.

Rule 16: Give-way The vessel shall take action to keep clear of the stand-on vessel.

Rule 17: Stand-on The vessel shall not take any action and keep its course and speed unless actions by the give-way vessel is not sufficient to avoid collision.

In addition, if a vessel is in neither of the situations, it has no special obligations regarding other vessels. An illustration of the different situations is shown in Figure 2.1.

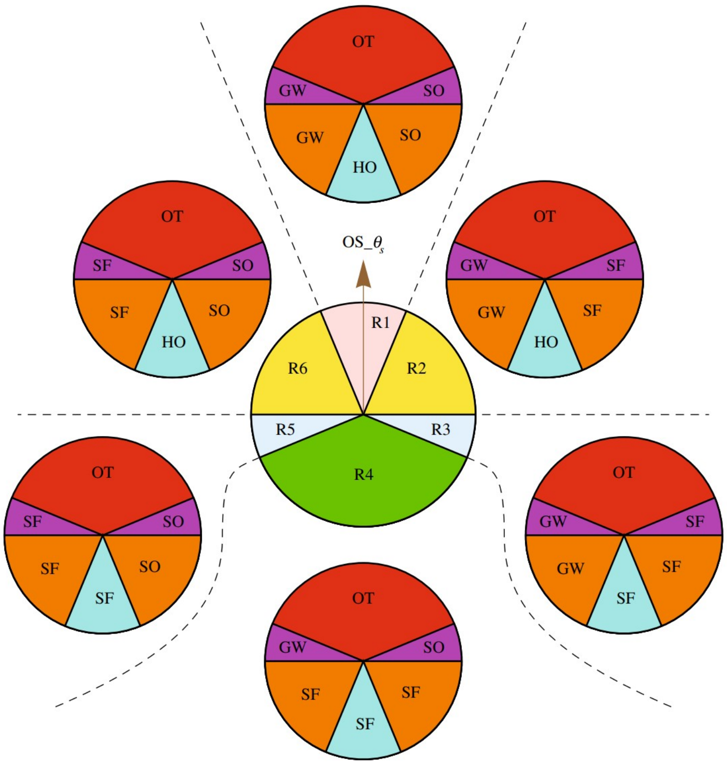


Figure 2.2: Illustration for the COLREG classification algorithm. The circle used is determined by the relative bearing of Ownship (OS) and Target ship (TS). The sector within that circle used is determined by the relative course of OS and TS. Figure courtesy of [Tam and Bucknall (2010)]

To act in accordance with the COLREGs in a vessel to vessel encounter, the rules that apply for that encounter must be determined. In the COLREGs, four encounter types are defined, where in each encounter, one of the four rules Rule 13-15 and Rule 17 apply. If the position and course of both vessels in the encounter is known, the COLREGs describe the criteria for classifying the encounter. Several algorithms for determining the COLREG situation exist, one of which is described in [Tam and Bucknall (2010)].

The method consists of two steps, with a graphical interpretation as shown in Figure 2.2.

Firstly, the relative bearing of the TS from the OS is applied to determine which relative bearing sector (R1-R6) the TS is within. The relative bearing φ is given by

$$\varphi = \mathrm{atan2}(E_{TS} - E_{OS}, N_{TS} - N_{OS}) - \chi_{OS}, \tag{2.1}$$

where χ_{OS} is the course of OS, and E_x and N_x are the east and north position of the OS and TS.

Subsequently, the situation sector is determined based on the course of the TS relative to the course of the OS. The relative course χ_{rel} is defined by

$$\chi_{rel} = \chi_{TS} - \chi_{OS}, \tag{2.2}$$

where χ_{TS} is the course of TS.

Figure 2.2 shows how these two angles are used. The relative bearing determines the sector around the OS, and the relative course determines the sector within the circle in the given bearing sector. Each course sector corresponds to a COLREG situation. Implementation of the method can be done easily through a lookup table, where the two angles in Equation (2.1) and Equation (2.2) are the lookup parameters.

2.2 Graph Search Algorithms

2.2.1 Dijkstra's Shortest Path Algorithm

Dijkstra's shortest path algorithm is a standard algorithm for finding the shortest path between nodes in a graph [Dijkstra (1959)]. It is a classic use of the greedy approach in computer science. The output of Dijkstra's algorithm is a search tree which describes the shortest path from the start node to each other node in the graph.

The graph is initialized with all nodes having a distance of infinity from the start node, except the start node which naturally is a distance of zero away from itself. All the nodes in the graph are added to a min-priority queue. At each step in the algorithm the node with the smallest distance is extracted from the priority queue. This is now the current node u. All the distances from the start node to each of the neighbors of u is then updated if travelling through u provides a shorter path to that neighbor. This iteration continues until the priority queue is empty. Pseudocode for Dijkstra's algorithm is given in Algorithm 1.

If one is only interested in finding the shortest path between two nodes, the algorithm can be stopped when u is the goal node. Because u is now the unexpanded node with the shortest distance and all distances between nodes are positive there is no way for the distance to u to get any shorter. One can thus conclude that the current path to u is optimal.

One may also add a heuristic to aid in the graph search. This is what is called the A* (A-star) algorithm [Hart et al. (1968)]. The heuristic may be distance to the goal measured as a straight line instead of by following the edges of the graph. This type of heuristic will help the algorithm in choosing to expand nodes that are more likely to lead to the goal. This may lead to the algorithm expanding fewer nodes before finding the shortest path to the goal, and thus decrease the run time of the algorithm. For the algorithm to still be optimal after adding this heuristic, the heuristic must be admissible [Korf (2000)]. That is, it does not overestimate the cost of reaching the goal.

Algorithm 1: Dijkstra's shortest path algorithm

```
in  : Source $s$, Graph $G$
out: Shortest path tree giving the shortest path from $s$ to all other nodes in $G$
$Q \leftarrow \emptyset$;
for vertex $v \in G$ do
    $v$.distance = $\infty$;
    $v$.parent = null;
    $Q$.add($v$);
end
$s$.distance = 0;
$S \leftarrow \emptyset$;
while $Q \neq \emptyset$ do
    $u \leftarrow Q$.extract_min();
    $S$.add($u$);
    for neighbor $v$ of $u$ do
        if $u$.distance + length($u$, $v$) < $v$.distance then
            $v$.distance = $u$.distance + length($u$, $v$);
            $v$.parent = $u$;
        end
    end
end
return $S$;
```

2.3 Geometry

In the proposed method a graph is built in three-dimensional space. Some geometric calculations are needed to find nodes to add to the graph and to check for intersections with obstacles.

2.3.1 Distance From Point To Line Segment

A line segment is determined by the equation $\mathbf{s} + t \cdot \mathbf{d}$, where $t_{min} \leq t \leq t_{max}$, $\mathbf{s}$ is a point on the line and $\mathbf{d}$ is the direction of the line.

The point on the line segment closest to a point $\mathbf{p}$ can be found using the following method:

$$t_{line} = \frac{(\mathbf{p} - \mathbf{s}) \cdot \mathbf{d}}{\mathbf{d} \cdot \mathbf{d}} \tag{2.3}$$

$$t_{segment} = \max(t_{min}, \min(t_{max}, t_{line})) \tag{2.4}$$

$$c = \mathbf{s} + t_{segment} \cdot \mathbf{d} \tag{2.5}$$

2.3.2 Intersection Between Line And Cone

The intersection between the cone and a line can be found by solving a second degree polynomial. For a line given by a point **p** and a direction **d**, and a cone with apex **a** and an opening angle of θ, the coefficients c_1, c_2 and c_3 of the polynomial can be found in the following way:

$$\boldsymbol{\Delta} = \mathbf{p} - \mathbf{a} \tag{2.6a}$$

$$\mathbf{u} = [0, 0, 1]^T \tag{2.6b}$$

$$c_1 = (\mathbf{d} \cdot \mathbf{u})^2 - \cos^2\theta \tag{2.6c}$$

$$c_2 = 2\left((\mathbf{d} \cdot \mathbf{u})(\Delta \cdot \mathbf{u}) - \mathbf{d} \cdot \Delta \cos^2\theta\right) \tag{2.6d}$$

$$c_3 = (\Delta \cdot \mathbf{u})^2 - \Delta \cdot \Delta \cos^2\theta \tag{2.6e}$$

For a solution q to the polynomial with the given coefficients, the intersection point **i** can be found using the following equation:

$$\mathbf{i} = \mathbf{p} + q \cdot \mathbf{d} \tag{2.7}$$

This approach sometimes gives extra solutions, belonging to the mirror of the cone around its apex. These are easily found by checking whether the solution lies above or below the apex of the cone. One may also want to only check if the cone intersects with a segment of a line, This can also be achieved quite easily by checking if the line parameter q is within the line segment limits.

2.3.3 Intersection Between Line And Plane

Given a line determined by the point **s** and the direction **d**, and a plane determined by the point **p** and normal vector **n**, the intersection between them can be found using the following:

Algorithm 2: Line-plane intersection

```
$l \leftarrow \mathbf{d} \cdot \mathbf{n}$;
$m \leftarrow (\mathbf{p} - \mathbf{s}) \cdot \mathbf{n}$;
if $l = 0$ then
    // Line is paralell to plane, no intersection
    return $null$;
else
    // Line intersects at a single point
    $t \leftarrow \frac{m}{l}$;
    return $\mathbf{s} + t \cdot \mathbf{d}$;
end
```

2.3.4 Intersection Between Line And Polyhedron

To check if a line intersects with a polyhedron one can check if the line intersects with each of the planes making up the faces of the polyhedron. When finding the intersection between a line and a plane the time at which the line intersects the plane is calculated. Using the fact that a line must enter the polyhedron before it exits, it can be determined if it is possible for the line to intersect the polyhedron. A Python implementation of this is given in Code listing 2.1.

Code listing 2.1: Function for checking for intersections for line and polyhedron

```python
import numpy as np
def line_polyhedron_intersect(polyhedron: Polyhedron, line: Line) -> bool:
    enter_time = line.start_time
    leave_time = line.end_time
    for face, normal in zip(polyhedron.faces, polyhedron.normals):
        # face is a list of vertices
        N = -float(np.dot(line.start - face[0], normal))
        D = float(np.dot(line.direction, normal))
        if D == 0:
            # Parallel
            if N <= 0:
                # Outside obstacle or contained in face, cannot intersect
                return False
            else:
                # Cannot enter or leave through this face, ignore
                continue
        intersection_time = N / D
        if D < 0:
            # Entering
            enter_time = max(enter_time, intersection_time)
            if enter_time > leave_time:
                # Enter after leaving, impossible
                return False
        else:
            # Leaving
            leave_time = min(leave_time, intersection_time)
            if leave_time < enter_time:
                # Leaves before entering, impossible
                return False

    # None if point at time is outside of line segment
    intersection_points = [line.point_at_time(enter_time), line.point_at_time(leave_time)]
    return any(
        lambda point: point is not None,
        intersection_points
    ))
```

2.4 Velocity Obstacle Method

Velocity Obstacle (VO) is a common method used for COLAV. A velocity obstacle is the set of velocities which will lead to a collision, given that the obstacle keeps a constant speed and heading. The basic idea of the method is to choose a velocity outside of any velocity obstacles and thus avoid colliding with any obstacle. [Fiorini and Shiller (1998)]

Each obstacle is represented by a circle. The circle is chosen to be large enough so that it incorporates the size of the controlled vessel in addition to the obstacle. In this way a single point can be used to represent the controlled vessel.

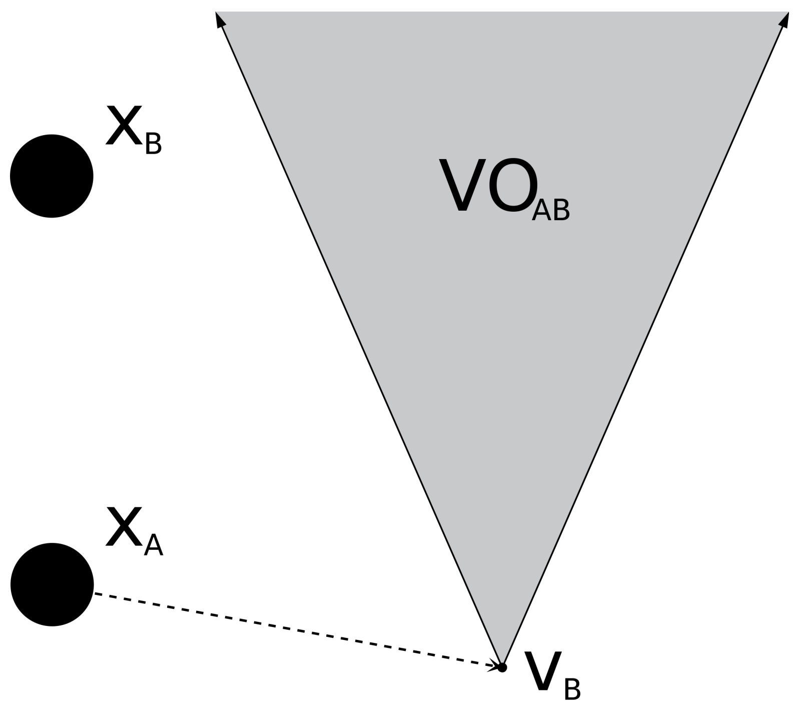


Figure 2.3: The velocity obstacle VO_{AB} for a vessel A at position X_A and a vessel B at position X_B with velocity V_B. Courtesy of Aubergine, CC BY-SA 3.0 https://creativecommons.org/licenses/by-sa/3.0, via Wikimedia Commons

[Fiorini and Shiller (1998)] first defines the *collision cone* for the controlled vessel A and the obstacle B as

$$CC_{A,B} = \{\mathbf{v}_{A,B} | \lambda_{A,B} \cap B \neq \emptyset\}, \tag{2.8}$$

where $\mathbf{v}_{A,B}$ is the relative velocity of A and B, $\lambda_{A,B}$ is the line of $\mathbf{v}_{A,B}$. The velocity obstacle is obtained by translating the collision cone by the velocity of the obstacle, $\mathbf{v}_B$. An illustration of this can be seen in Figure 2.3.

The controlled vessel has a desired velocity $\mathbf{v}_{desired}$. This velocity is commonly pointing towards a goal position or a waypoint. $\mathbf{v}_{desired}$ may be inside the velocity obstacle of one of the obstacles, and can as such not be used. Instead, the velocity $\mathbf{v}_{chosen}$ that is outside any velocity obstacle and closest to $\mathbf{v}_{desired}$ is chosen. In this way the controlled vessel will avoid collisions while making the minimal changes necessary and can efficiently reach its goal.

Chapter 3

An Area-Time Approach To Collision Avoidance

In this chapter, the proposed algorithm for trajectory planning is described. Firstly, an overview of the algorithm is given. Subsequently, it is described in detail how the working space is constructed and how obstacles and other features are represented in it. Then, the steps to build and search the graph, as well as methods for continuous validation of the trajectory and replanning is explained.

3.1 Overview

The principles of the method is to construct a three dimensional space, spanned by the admissible area, that is the area the vessel is allowed to operate in, and time. This is called the working space and is denoted by $\mathcal{W}$. In the working space, relevant features of the environment, such as moving vessels and static obstacles, can be represented as stationary volumes. The admissible area is a continuous area that contains both the start position and the destination, and is by design free of static obstacles. The task of collision free trajectory planning is then reduced to a path-planning problem in the three dimensional space. The method of representing the problem in $\mathcal{W}$ also provides an intuitive framework for visualizing and understanding the problem.

The method generally consists of four steps:

1. Transform relevant features into the working space $\mathcal{W}$.
    - Dynamic obstacles
    - The admissible area
    - Start position
    - Goal position
2. Build a graph that represents a set of collision-free paths from the start position to the goal position.
3. Search the graph to find the minimum cost path.

4. Extract temporal waypoints from the minimum cost path and construct a trajectory from the waypoints.

In the development of this method, the following assumptions are made:

Assumption 3.1. The space from the start position to the goal position is connected and is thus traversable.

The space has to be traversable. This means that there exists a path from the start position to the goal position within the admissible area that is not permanently blocked by any static or dynamic obstacles. This is a reasonable assumption for this method and it is fair to assume that if such a situation were to occur, it would be handled by a higher level planner in the autonomy pipeline.

Assumption 3.2. There is enough space for all vessels to move about.

The situation is not so crowded that there is need for a traffic separation scheme or other higher level cooperation between vessels.

Assumption 3.3. The position and velocity of all dynamic obstacles are known.

It is assumed that the vessel's situational awareness system is capable of detecting all relevant obstacles either through exteroceptive sensors or through communication systems such as Automatic Identification System (AIS). This is a somewhat weak assumption, since most small vessels and recreational vessels lack AIS equipment, and the field of view of exteroceptive sensors can be restricted in confined space by for example static obstacles. Functionality for validating the current trajectory based on updated information is implemented. This way, some of the effects of the missing information can be mitigated.

Assumption 3.4. All dynamic obstacles move with constant speed and heading.

When representing dynamic obstacles in the working space, it is assumed that all dynamic obstacles move with a constant speed and course. This is not a reasonable assumption in confined waters, as constant speed and heading for an extended amount of time can be reckless due to the presence of static obstacles. However, in such confined waters, there is often a pattern for how vessel maneuver, which can be used to make a more qualified prediction of the future behavior of dynamic obstacles. In the proposed graph-building method functionality for waypoints for dynamic obstacles is implemented, where it is assumed that the target-ship keep a constant course and speed between waypoints.

3.2 The admissible area

The admissible area is the area that the ASV is allowed to operate in. It must be constructed such that there are no static obstacles in it and the planner can choose any trajectory contained within it.

The admissible area would typically be chosen by humans with the aid of a nautical chart. A safety margin should be added to all features in the area so that the admissible area is smaller than the physical area it is possible for the ASV to move in. Because the trajectory planner is allowed to create a waypoint at the edge of the admissible area, the safety margin should also incorporate any expected tracking error.

In this thesis the admissible area is represented as a simple, connected polygon. It is desired that the polygon does not include every detail of the environment, to reduce the amount of calculation needed to find intersections with the border. This fits well together with the safety margin, which will smooth out small details.

3.3 The Nominal Path

The nominal path is a waypoint mission from the start position to the goal position that is assumed to exist. It may be created by a human or a higher level planning algorithm. For ASV operations that are repetitive, such as ferry operations or local cargo distribution, a set of nominal paths along the transit routes can easily be constructed manually. Alternatively, algorithms such as the visibility-graph[Lozano-Pérez and Wesley (1979)] can be used to calculate a trajectory that is collision free with static obstacles.

The nominal path is used to guide the method in the correct direction. If there are no dynamic obstacles, the controlled vessel should follow the nominal path. This is incorporated into the cost function, as described in Section 3.8. The nominal path is also used to create a heuristic that is used during graph building. This addition speeds up the graph building considerably by favouring nodes that are closer to the goal position.

3.4 Obstacle Representation

In the xy-plane, the obstacles are represented as rhombuses. Rhombuses serve as a simple approximation for ship shapes. Rhombuses are simple, convex polygons, which makes geometric calculations easier and faster than they would be with a more complex obstacle representation.

The length and width of the rhombuses must be chosen larger than the obstacle itself. In addition to the size of the obstacle itself, the obstacle representation must incorporate the size of the controlled vessel, any tracking error and any desired safety margin. By doing this, the controlled vessel can be represented as a point without extension. This larger rhombus is called the safety region and is used as the obstacle representation when building the graph. An illustration of the obstacle representation together with the obstacle is shown in Figure 3.1.

3.5 Representing Features in Area-Time Space

When transforming features from the xy-plane to area-time space, they gain an additional dimension, namely the time dimension. Stationary points, like the vertices of the border

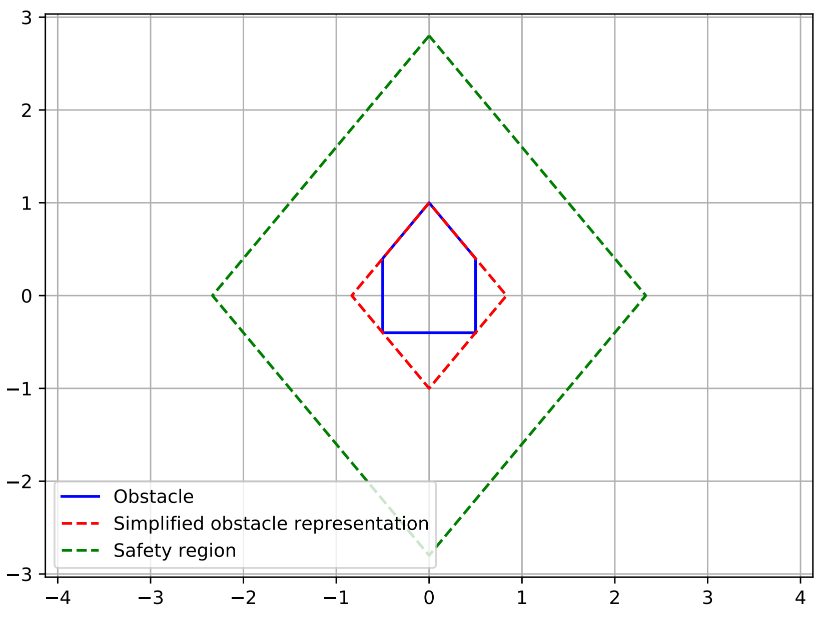


Figure 3.1: A dynamic obstacle representing a marine vessel with the simplified obstacle representation and the safety region.

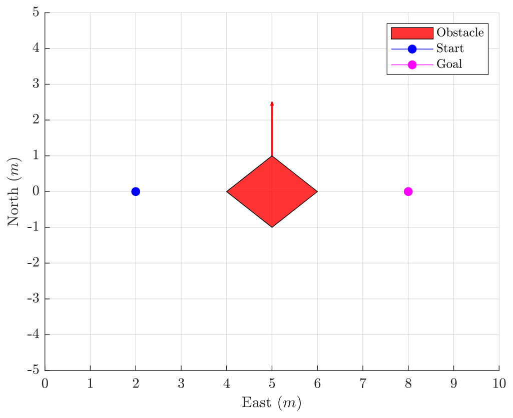
around the working area or the goal position, become vertical lines. Non-stationary points may become an arbitrarily complex curve in area-time space. However, if they move with constant velocity (Assumption 3.4), they become slanted lines. A point starting at $[x_0, y_0]$ at time t_0 moving with constant velocity $[v_x, v_y]$ will form the following line:

$$\begin{bmatrix} x_0 \\ y_0 \\ t_0 \end{bmatrix} + t \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix}, \quad t \geq t_0. \tag{3.1}$$

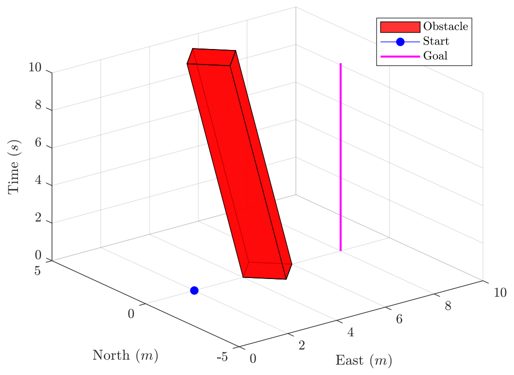
Here, t is some time after t_0.

Assuming obstacles move with constant velocity and heading (Assumption 3.4), an obstacle transformed into area-time space becomes an oblique rhombic prism. By transforming each of the vertices of the rhombus from the xy-plane to area-time space, four lines are obtained. These lines form four of the edges of the prism, connecting the base faces. A comparison of the different features in the xy-plane and in area-time space can be seen in Figure 3.2.

If there exist better estimates for how an obstacle will move in the future and a waypoint mission for this obstacle can be made, the obstacle representation can be improved. A waypoint mission for the obstacle can be represented by considering each leg of the mission as a separate obstacle that only exists for some interval of time. By doing this, obstacles where a prediction of the intent exist can be transformed by the same methods as for obstacles where a constant behavior is assumed.



(a) Three different features in $\mathbb{R}^2$ representing the area. The red polygon is the obstacle representation, where its velocity is denoted by the red arrow.



(b) The different features transformed into $\mathcal{W}$.

Figure 3.2: The start position, the goal position and an obstacle transformed into $\mathcal{W}$

3.6 Building the Graph

When building the graph a similar approach to that of Dijkstra's algorithm is used. At the start the graph only consists of a single node, representing the start position. In each iteration, the node representing the lowest cost path so far is chosen to be expanded. When expanding a node, first a set of reachable nodes are found. These represent possible movements for the vessel, ignoring any obstacles. Then, each edge from the current node to the new nodes are checked for collisions. Nodes that create an edge with collisions are filtered out. For each of the remaining nodes the cost function and a heuristic is evaluated. Combined, the cost and the heuristic is used to determine the next node to be expanded. This continues until a sufficient number of paths to the goal is found.

Algorithm 3: Graph-building algorithm

```
input     : Nominal path waypoints $\mathcal{N} = n_0, n_1, \ldots, n_n$
            Obstacle prisms $\mathcal{O}$
            Border walls $\mathcal{B}$
parameter: Minimum number of goal nodes to create $N_{paths}$
output    : Graph representing a set of collision free paths from start to goal

$G \leftarrow \{\text{Node}(n_0)\}$
$N_{goal} \leftarrow 0$
while $N_{goal} < N_{paths}$ do
    $m \leftarrow$ extract_min $(G)$
    $\mathcal{R} \leftarrow$ reachable $(\mathcal{O}, \mathcal{B}, \mathcal{N}, m)$
    $\mathcal{C} \leftarrow \{r \in \mathcal{R} \mid r \text{ is collision free}\}$
    $G \leftarrow G \cup \{\text{Node}(c, \text{parent}{=}m) \mid c \in \mathcal{C}\}$
    $N_{goal} \leftarrow N_{goal} + \sum_{n \in G} [n \text{ is on } n_n]$
end
return $G$
```

3.6.1 Finding Reachable Nodes

To find a set of reachable points for the vessel, a set of feasible speeds $\mathcal{S}$ is chosen. All the positions that are possible to reach by using a constant speed and heading form a cone with the apex at the current position of the vessel. Such a cone is constructed for each of the speeds in $\mathcal{S}$. Because there is an infinite number of points the vessel can move to, a finite subset has to be chosen. The subset $\mathcal{I}$ chosen is the set of all the points where the cone intersects with any of the following features in the area-time space:

- The line representing the goal position
- The lines at each of the waypoints along the nominal path
- The lines representing the vertices of the working area
- The edges of all the obstacles

All of these intersections can be found by using the method described in Section 2.3.2.

It is desired that the controlled vessel keeps a distance to the obstacles if there is room to do so. By using the edges of the obstacle representation to generate nodes, the controlled vessel is likely to move close to the safety region. To combat this, the obstacle prism is scaled linearly with the distance from OS to the closest obstacle. The scaling factor is bounded by 1.0 and some maximum scaling, in this thesis set to 1.5. This will give the generated trajectories larger safety margins, which is useful in case of any estimation error in the position or velocity of the obstacle. It will also create a more intuitive trajectory. A human operator would not choose to pass an obstacle as close as possible, but rather keep a distance.

A limited time horizon is implemented. Because there is uncertainty in the position and velocity of the obstacles, trajectories based on this information far into the future

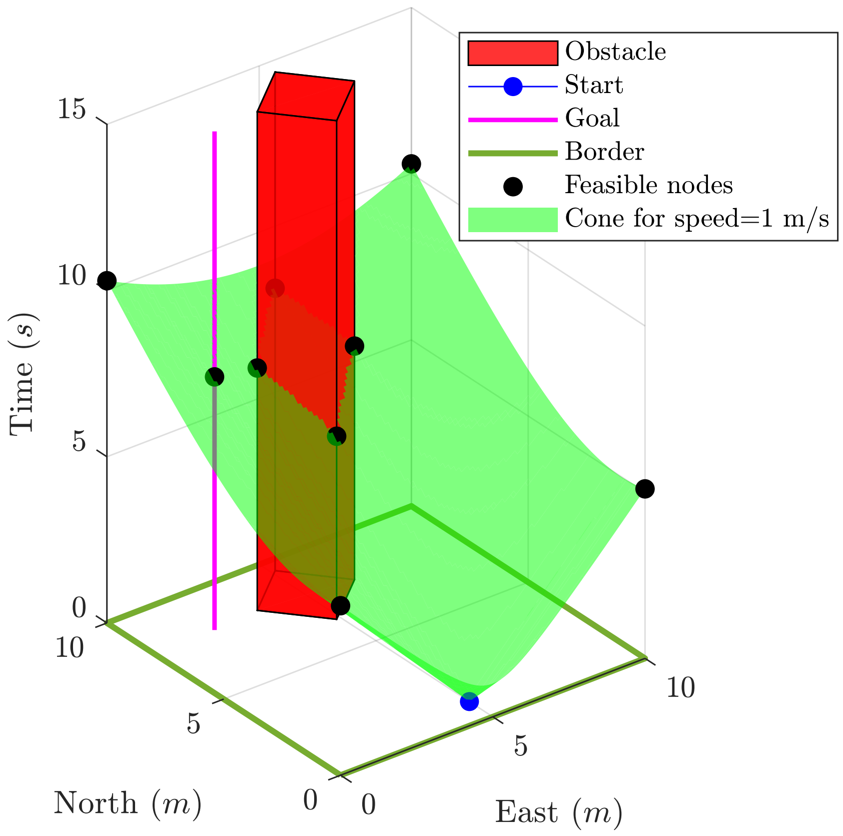


Figure 3.3: Illustration of the intersection of a speed cone and features in $\mathcal{W}$.

are likely to be wrong and a replan would be necessary. The time horizon is implemented by having the obstacles only exist for a limited time into the future. No nodes are thus added on the obstacle prisms after the time horizon.

In addition to speed and heading maneuvers, it is desirable that the vessel can stop and wait. All the positions in area-time space that are possible to reach by standing still make up a vertical line in area-time space. To facilitate standing still, a set of waiting intervals $\mathcal{T}$ is chosen. For each of the waiting intervals in $\mathcal{T}$ a node on this vertical line is generated. $\mathcal{H}$ is the set of these nodes.

The set of all reachable nodes $\mathcal{R}$ is the union of $\mathcal{I}$ and $\mathcal{H}$.

3.6.2 Eliminating Edges With Collisions

Each of the edges from the current node to each of the reachable nodes may intersect with an obstacle or the boundary of the working area. An edge is intersection free if the line segment from the current node to the reachable node does not intersect with any of the obstacles or cross the boundary of the working area. In addition to being free of intersections, an edge must be contained within the working area to be collision free.

The set of collision free nodes $\mathcal{C}$ is the subset of $\mathcal{R}$ for which the edge between the node and the current node is intersection free and is contained entirely within $\mathcal{W}$

Collision detection can be done by checking for intersections between the line segment representing an edge in the graph and each face of each obstacle and each face of the border of the admissible area. However, this poses some problems. First of all, there are some special cases to take care of, for example if the admissible area is not convex. If each end of a line segment is on the surface of an obstacle prism, a similar problem occurs where there are no intersections, while the edge is not collision free. These cases can be fixed by checking whether the midpoint of the line segment is inside or outside the obstacle or admissible area. Algorithms exists for this, and they are commonly used in computer graphics to check if a point is on the inside or the outside of a polygon. Two common approaches for this are the “odd-even” or “winding number” rules [Bellamy-Royds et al. (2020)].

A second problem is numerics. For example, when a line segment intersects with an obstacle prism at the edge or a line segment is contained in a face of an obstacle prism, small numerical differences can change the result. These edge-cases have to be handled. In this thesis this problem is solved by tracking some additional information about which face of which obstacle or which border segment the intersection lies on. Given this information one can disambiguate between different cases. In addition, upscaled versions of the obstacle representation of dynamic obstacles are used to generate nodes if there is room, these cases become less frequent. Without this scaling, nodes would always intersect with an obstacle prism and numerical errors would become much more apparent.

3.6.3 Stop Condition

Because the graph could grow infinitely large, there needs to be a condition for when the building of the graph should stop. There are at least two different ways to do this. The first method is to create N_{paths} different paths from the start position to the goal position and then choose the best one. The second method is to create paths to the goal position until a sufficiently low cost path is created. This path is then chosen. In this thesis the first approach is used. There are benefits to both approaches. The second method will guarantee that a sufficiently low cost path is chosen, given that it exists. However, such a path may not exist. In that case, the graph building algorithm will never terminate. The first approach will always terminate, given that Assumption 3.1 and Assumption 3.2 holds. N_{paths} does not need to be large. During the development of the method it was observed that the first few paths that were built were the simplest and the later paths were more complex variations of the first paths. In general the simpler paths are also safer, because they are more intuitive and communicates the intention of the vessel clearer to other seafarers. It was observed that these first built paths were often the ones with the lowest cost and were thus chosen.

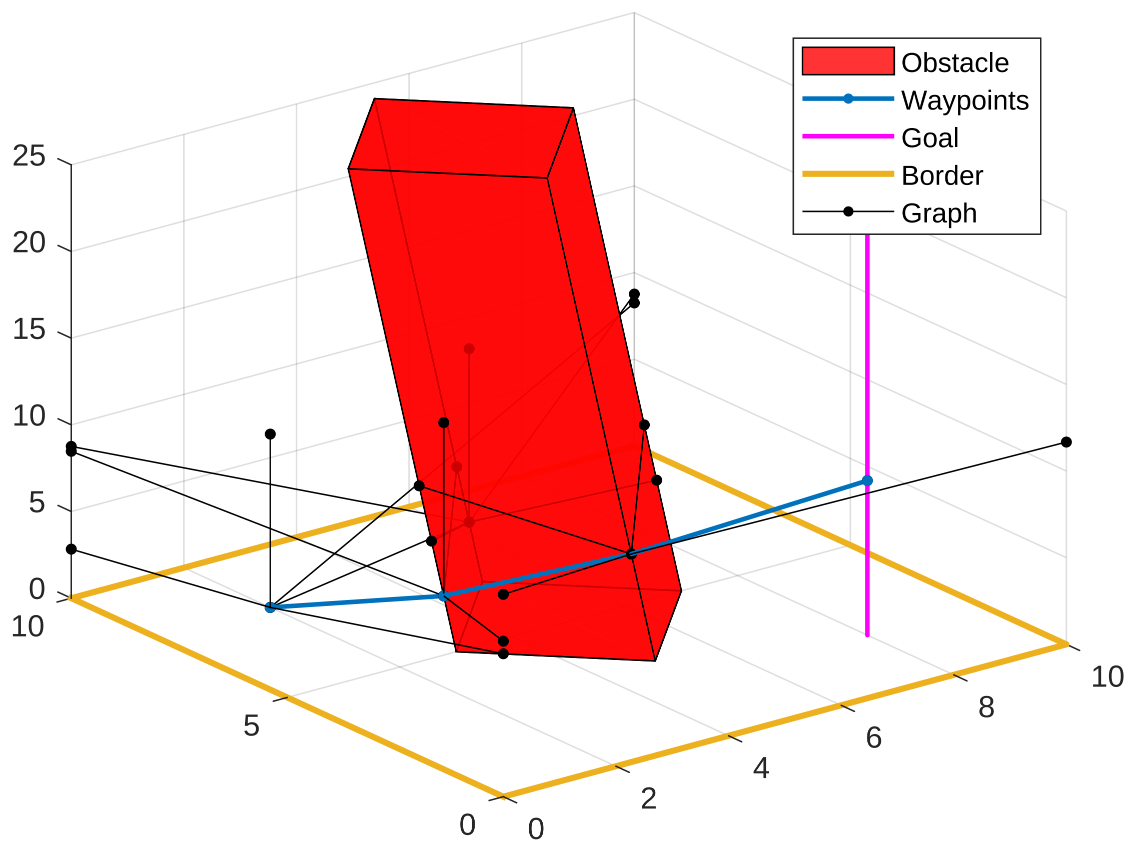


Figure 3.4: A small example graph created by the graph building method. The lowest cost trajectory is marked in blue.

3.7 Searching the Graph

When building the graph, N_{paths} paths to the goal is generated, representing different trajectories for reaching the goal position. Each of the paths must be evaluated to find the best one. Fortunately, most of the work is already done. Because the graph built is a tree, there is only one way to reach each node. This means that the cost can be summed up and stored in each of the nodes during the building of the graph. The search function thus only has to loop over the goal nodes and choose the one with the lowest cost.

To get the actual waypoint mission out of this, the tree is walked from the lowest cost node at the destination to the root. The root node is the node representing the start position, which by design is at the current position of the controlled vessel. Each node along the lowest cost path is then a waypoint in area-time space, which can be interpreted as a waypoint with a timestamp in $\mathbb{R}^2$. From this set of waypoint, a trajectory with constant course and velocity between waypoints can be constructed. The resulting trajectory will have steps in both velocity and heading at each waypoint, and some post processing of the trajectory should be applied, to produce smooth references for the vessel control system.

3.8 Cost Function

To determine which of the trajectory candidates in the graph is best, a cost function is needed.

By applying an appropriately formulated cost function, a relevant cost can be calculated for each new node in the graph building process, and hence improve the chances of extending the graph along the best candidate paths. However, designing such a cost function is not a trivial task, as the metrics for describing a trajectory candidate that is good with respect to the task can be hard. Examples of considerations that should be included in the cost function are:

Safety with respect to static and dynamic obstacles This can typically be represented through the distance to the obstacle, either in space or time.

COLREG The trajectory should comply with the relevant rules from the COLREG, and therefore this should be encoded in the trajectory cost.

Distance The length of the trajectory should not be longer than necessary, to avoid both long transit times and high energy usage.

Area What kind of area the trajectory is passing through. Is it along the correct side of a canal, or across a highly trafficked area?

Good seamanship Will the trajectory give a transit that is in compliance with good seamanship. Is it smooth with a velocity that is in line with what the surroundings dictate. Is it predictable for other vessels in the vicinity.

Some of these considerations are easy to encode in a cost function, whereas other are more vague and abstract. One also has to consider how these considerations interact with each other, and which to give the most weight to.

3.8.1 Keeping Distance To Obstacles

Assigning a penalty to trajectories that are close to other vessels is fairly straight forward. At each point in time, each obstacle is represented as a two dimensional polygon. The minimum distance from the vessel to each obstacle at a given point in time is found by finding the shortest distance from the vessel to each of the line segments of the polygons representing the obstacles. The distance from the vessel to each line segment can be found as shown in Section 2.3.1.

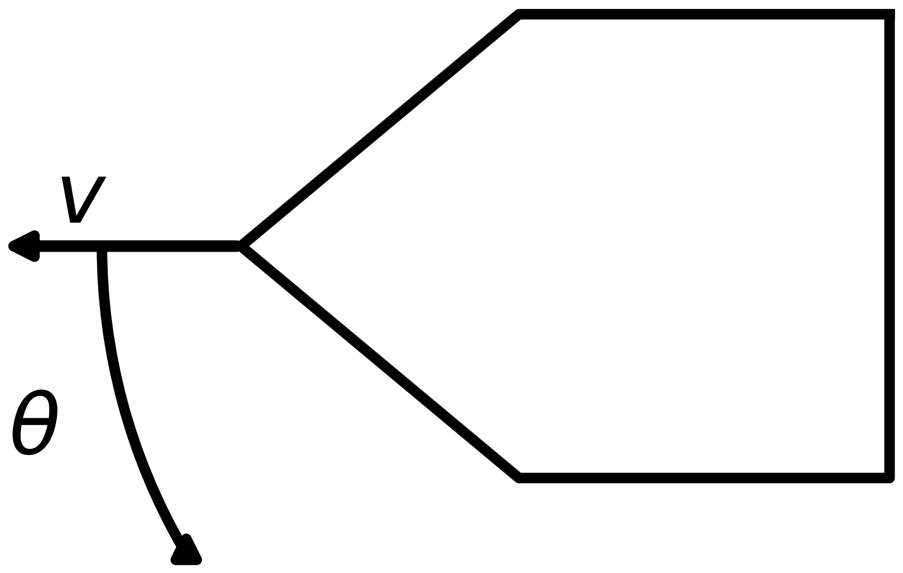


Figure 3.5: An obstacle with velocity v and the definition of θ marked.

3.8.2 Following COLREG

Designing a cost function that honors all of the rules in COLREG is hard. The rules are not as concrete as distances between objects. A phrase from the COLREG is that any action taken to avoid collision should be "with due regard to the observance of good seamanship", which one cannot to describe mathematically. However, COLREG rules 13-17 defines a set of situations and which actions the vessels in the encounter should take in each situation. The method used for classification in this thesis is the one presented in Section 2.1.

In most of the situations the action to take is to maneuver to a certain side of the other vessel. For example, in a head on encounter, the two vessels shall maneuver to starboard so they pass each other port to port. Rules such as these are possible to encode mathematically.

The angle θ is defined as the relative bearing between the OS and TS. Different sectors with high and low cost can then be defined in terms of this angle. A function with period 2π with a high value and low value region is wanted. It is also necessary to be able to rotate the high and low cost sectors around the obstacle to account for the different situations defined by COLREG. A function that fulfills these requirements is

$$c(\theta, \alpha) = (1 + \cos(\theta + \alpha))^2 . \tag{3.2}$$

Here, α is used to rotate the high and low cost regions around the obstacle so that the cost represent the COLREG compliance. The values of α for different classifications is given in Table 3.1.

In the stand-on and safe situations it is assumed that no special action is required. In the stand-on situation this is because it is the obstacle which is supposed to take actions,

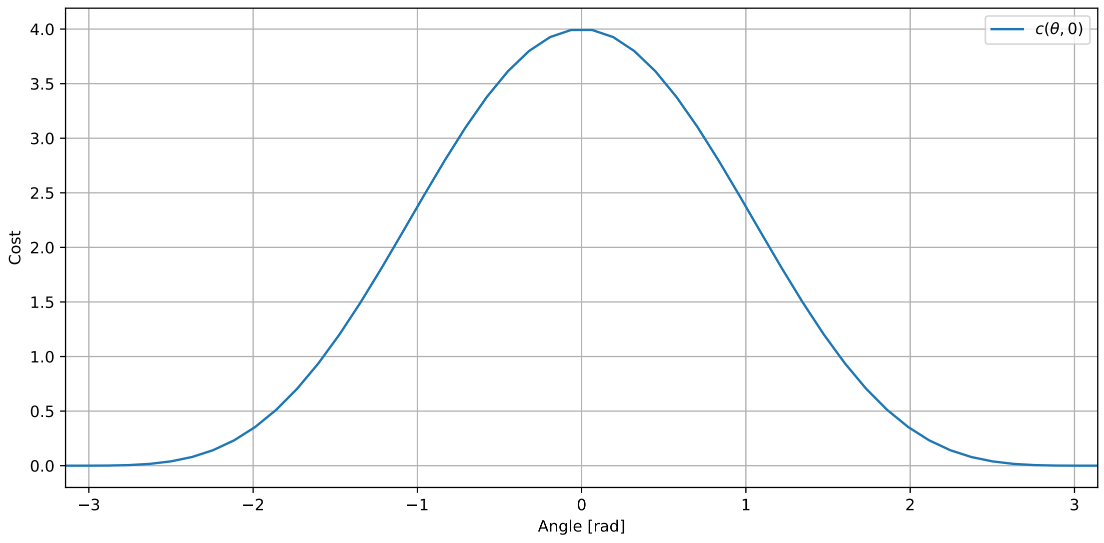


Figure 3.6: A plot of $c(\theta, \alpha)$ where $\alpha = 0$. It is clear that moving in front of the obstacle has a high cost.

as it is the give-way vessel. The method will still create a trajectory that does not collide with the estimated path of the obstacle, but no cost is added to avoid certain sides of the obstacle. If it turns out that the planned trajectory will lead to a collision, a replan i triggered.

In the safe situation the vessels are moving away from each other, so no action should be required.

Table 3.1: Values for α in the different COLREG situations

Situation	α
Safe	-
Stand on	-
Give way	0
Head on	$\pi/4$
Overtaking port	$\pi/4$
Overtaking starboard	$-\pi/4$

3.8.3 Keeping To The Nominal Path

As previously mentioned, it is assumed that there exist a nominal path to the desired destination that is free from collisions with static obstacles. It is further assumed that the quality of this path is such that if no obstacles exist, or are in conflict with traveling along that path, it is desirable to follow the nominal path. To facilitate trajectories that travel close to, or along the nominal path, a cost on the distance to the nominal path is included. The distance to the nominal path can be calculated by finding the minimum distance to each of its segments. The distance to each segment can be calculated as shown in Section 2.3.1.

3.8.4 The Complete Cost Function

The three different cost functions presented capture different aspects of the considerations listed at the start of the section. The complete cost function at a point is the summation of these. To find a good evaluation of the cost of travelling along an edge of the graph, the cost can not simply be calculated in a single point. Instead, it should be integrated along the edge of the graph. A closed form solution for this integral is likely to be complicated, so instead the cost function is sampled along the edge at a fixed time interval.

In addition to the costs mentioned, a small cost for the length of the trajectory was added. This addition penalizes complicated trajectories that does not move towards the destination in a reasonable manner. Simpler trajectories tend to be more intuitive and closer to what a human would expect. By being more predictable for other seafarers, safety is increased.

Pseudo code for evaluating the cost of an edge is given in Algorithm 4.

Algorithm 4: Algorithm for evaluating cost along an edge

```
input : Nominal path waypoints $\mathcal{N} = n_0, n_1, \dots, n_n$
        Obstacle prisms $\mathcal{O}$
        Edge start $\mathbf{s}$
        Edge end $\mathbf{e}$
$C \leftarrow k_{travelled} \cdot |\mathbf{e} - \mathbf{s}|$
for $\mathbf{p}$ on line between $\mathbf{s}$ and $\mathbf{e}$ with interval $t_{step}$ do
    for $o \in \mathcal{O}$ do
        $C \leftarrow C + \frac{k_{distance}}{\text{distance}(\mathbf{p},o)} + k_{colreg} \cdot \text{colreg}(\mathbf{p}, o) + k_{nominal} \cdot \text{distance}(\mathbf{p}, \mathcal{N})$
    end
end
return $C$
```

The four parameters $k_{travelled}$, $k_{distance}$, k_{colreg}, and $k_{nominal}$ control the size of the cost function and the relative weights of the different parts. The values were found using trial and error, and it was found that k_{colreg} should be the dominant parameter to get correct COLREG behavior.

3.9 Verification and Replanning

As new and updated information comes in, the original planned trajectory may no longer be collision free. Obstacles may not have behaved as expected or new obstacles may have been detected. Because of this, each time new information is received from the sensor system, the current trajectory is checked for collisions with the updated obstacles. This is done in the same way as described in Section 3.6.2. If a collision is detected, a replan is triggered and a new graph is built as described in Section 3.6. The new graph is built from the current position of the vessel, as opposed to the original starting position. The graph is searched and a new trajectory is generated as usual. This replanning step makes it possible to use the method described in this chapter even if not all obstacles are known beforehand. Because of the feedback loop created by the validation and replanning of the trajectory, the method becomes more robust.

3.10 Heuristic

In addition to the cost calculated for each edge in the graph, a heuristic is used for more sophisticated node expansion in order to reduce the size of the graph. The heuristic used is the distance to the goal along the nominal path.

For a point p and a nominal path defined by the waypoints $(w_1, w_2, \ldots, w_n)$ the heuristic is calculated in the following steps:

1. Find the point c_i on each segment of the nominal path that is closest to p. This can be done as shown in Section 2.3.1.
2. Choose c as the c_i that is closest to p. The index i gives the segment of the nominal path that p is closest to.
3. Calculate the heuristic h by summing up the distances along the nominal path:

$$h = |\overline{pc}| + |\overline{cw_{i+1}}| + \sum_{k=i+1}^{n-1} |\overline{w_k w_{k+1}}| \tag{3.3}$$

The distance from each waypoint on the nominal path to the goal is precomputed before the graph building starts.

Using this simple heuristic dramatically cuts down the amount of nodes needed to explore before finding a goal node. However, the heuristic is not guaranteed to be admissible. That means the first goal node to be found is not guaranteed to have the lowest cost. To remedy this, N_{paths} paths to the goal are built, before the lowest cost path is chosen. The parameter N_{paths} is a tradeoff between decreasing runtime and increasing confidence in finding a close to optimal path.

3.11 Runtime

The main factor affecting the runtime of the algorithm is the number of nodes that are explored during the building of the graph. For each node that is explored, a set of reachable nodes has to be found, and all the reachable nodes must be checked if they are collision free.

It will take longer to build a graph for a situation with more obstacles, more vertices in the admissible area and a more complex nominal path. This is because a node is generated for each speed in $\mathcal{S}$ for each edge on the obstacle prism, each vertex line and each waypoint on the nominal path.

Additionally, each reachable node must be checked for collisions. This requires checking for intersections with all obstacles and border segments.

When constructing the admissible area, one should thus not include all the small details in for example a coastline, but rather make an approximation that lies within the actual allowed admissible area.

The size of the admissible area does not affect the runtime. This is because the nodes are found by finding intersections between geometric objects, and not by iterating through the admissible area. The distances between the objects does not matter.

There are a few parameters that can be used to control the number of nodes that are created:

N_{paths} : The minimum number of goal nodes to create

$\mathcal{S}$: The size of the set of speeds and hence the number of speed cones used to create the set of reachable nodes.

$\mathcal{T}$: The size of the set of waiting times used to create the set of reachable nodes

For each of these parameters there is a tradeoff between runtime and the quality of the generated trajectory. If the sets $\mathcal{S}$ and $\mathcal{T}$ are too large, too many nodes will be created and the runtime will suffer. However, if the sets are too small, the maneuvers the vessel can make will be limited. A larger set of speeds and waiting time will give a more complete set of possible trajectories and facilitate a more predictable and intuitive vessel operation. If the magnitude of the cost function is large, it will dominate the heuristic, and the search will be less efficient. It is thus important to balance these so that the algorithm finishes the search in a reasonable time while also giving reasonably safe trajectories.

When running the simulations presented in Chapter 4, the average measured runtime for each planning is 484ms, and the longest measured runtime is 963ms. The simulations are run on an Intel Core i7-6500U CPU. These runtimes are within the expected planning frequency for a vessel. More complex situations may however lead to longer runtimes. A "planning" includes the building and searching of a graph. One simulation may have multiple plannings du to invalidation of the current trajectory due new information becoming available.

Chapter 4

Simulation Results

A set of simulations are presented, where the capacity of the proposed planner to plan trajectories in simple and complex environments is demonstrated. Results from two other planning methods are also presented, and their performance is compared to the proposed method.

Data from the simulations are presented through three kinds of figures:

1. A figure showing an overview of the situation, which contains:
   - The trajectory of all involved vessels.
   - A vessel representation at matching timesteps, at each 10th of the total simulation time. The vessel representation is omitted after it has reached its destination to make the figures simpler and easier to read.
   - Static obstacles, where they are relevant.
2. A figure showing the shortest distance from the OS to any TS.
3. A figure showing the speed profile for the trajectory

4.1 Simulation Setup

The simulations are run in a simulator-environment written in Python.

In the simulations, a model of the milliAmpere, a prototype vessel for fully electric autonomous passenger ferries, is used. The model parameters are determined through the work presented in [Pedersen (2019)].

In addition to the proposed method, SP-VP and Velocity Obstacle (VO) is run on the same situation for comparison. There already exists an implementation for SP-VP for use with the simulation framework, provided by Bjørn-Olav Holtung Eriksen. The VO implementation is based on the implementation provided in [Guo and Zavlanos (2018)]. Some changes has been made to make the implementation fit into the simulation framework. In compliance with the GPL license, the source code with the changes are provided on the Github page of the author (https://github.com/hermabe/RVO_Py_MAS).

The parameter values used are listed in Table 4.1.

Table 4.1: Parameter values used during the simulation

Parameter	Value	Unit
$k_{travelled}$	1	m^{-1}
$k_{distance}$	1	m
k_{colreg}	100	1
$k_{nominal}$	0.01	m^{-1}
n_{paths}	20	1
t_{step}	5	s
$\mathcal{S}$	$\{0.3, 0.5, 1.0\}$	$\mathrm{m\,s}^{-1}$
$\mathcal{T}$	$\{20.0\}$	s

4.2 Situation 1: Head On In Open Water

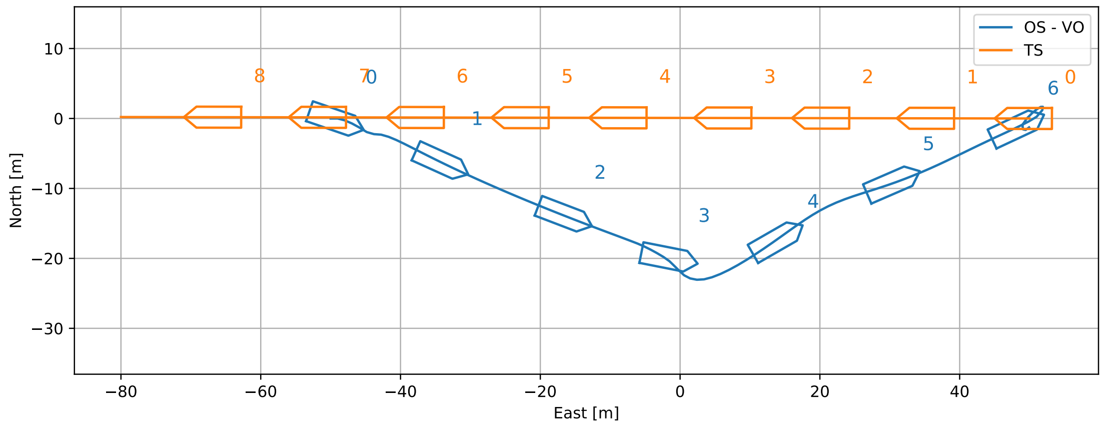
In this situation, the ASV meets another vessel in a head on encounter. To move past the dynamic obstacle the ASV must do a course changing maneuver, assuming that the oncoming vessel does not change its course. According to COLREG, the vessels should pass each other port to port, both doing a starboard course maneuver.

SP-VP is unable to do course maneuvers, so the method breaks down in this situation. It detects that the problem is infeasible and it cannot handle the situation alone. A system using SP-VP must have a higher level algorithm or a low-level reactive algorithm that can handle the encounter when such infeasibility occurs.

VO does a course changing maneuver as expected and avoids collision with the oncoming vessel as seen in Figure 4.1. VO has no concept of which action to take to comply with COLREG. VO simply chooses the smallest course change needed to avoid collision at each timestep. In this case however, it chooses to pass on the correct side of the oncoming vessel and passes it on the port side. There is no guarantee that this will work in general, and the algorithm may choose a side to pass on arbitrarily.

Figure 4.2 shows that AT handles the head on situation correctly. The OS passes the TS on the port side, in accordance with COLREG. The trajectory chosen by AT in this situation is similar to the one VO chose, but AT has methods in place to ensure that the chosen trajectory complies with COLREG and will create similar trajectories in similar situations.

When comparing the distance from OS to TS, the two algorithms perform similarly, as seen in Figure 4.3. The trajectory generated by AT is somewhat closer, but still outside the safety region. However, when comparing the speeds of the trajectories, there are some differences. Figure 4.4 shows that AT holds a more constant speed while VO varies the speed more. This can be attributed to the fact that VO can choose any velocity from a continuum, but AT must choose from a limited discrete set of speeds and headings. VO also becomes oscillatory when the destination is reached. When this happens, a higher level planner should take control and change OS into Dynamic Positioning (DP) mode or dock to a port.



S

Figure 4.1: Situation 1: Head on. The OS and TS in the xy plane. Trajectory generated by VO. TS is travelling from east to west while the OS travels from west to east and does a course maneuver to avoid collision with the TS.

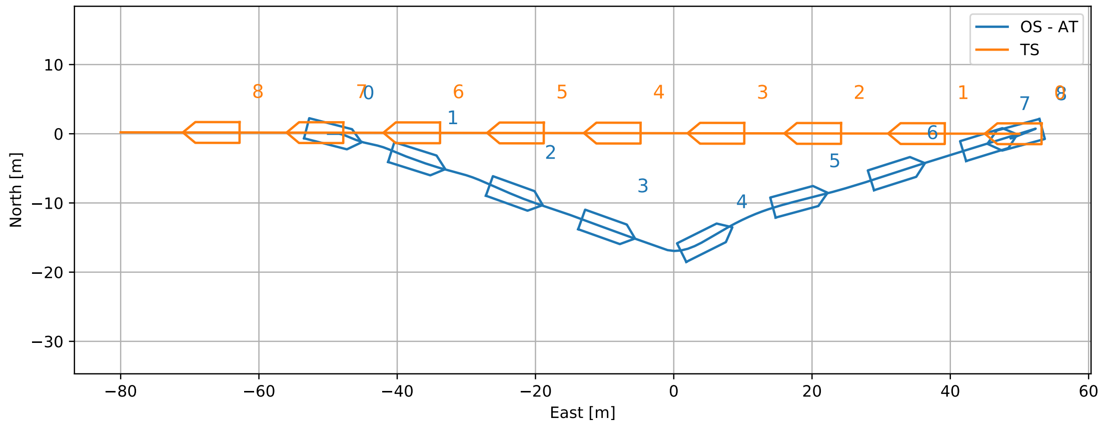


Figure 4.2: Situation 1: Head on. The OS and TS in the xy plane. Trajectory generated by AT. TS is travelling from east to west while the OS travels from west to east and does a course maneuver to avoid collision with the TS.

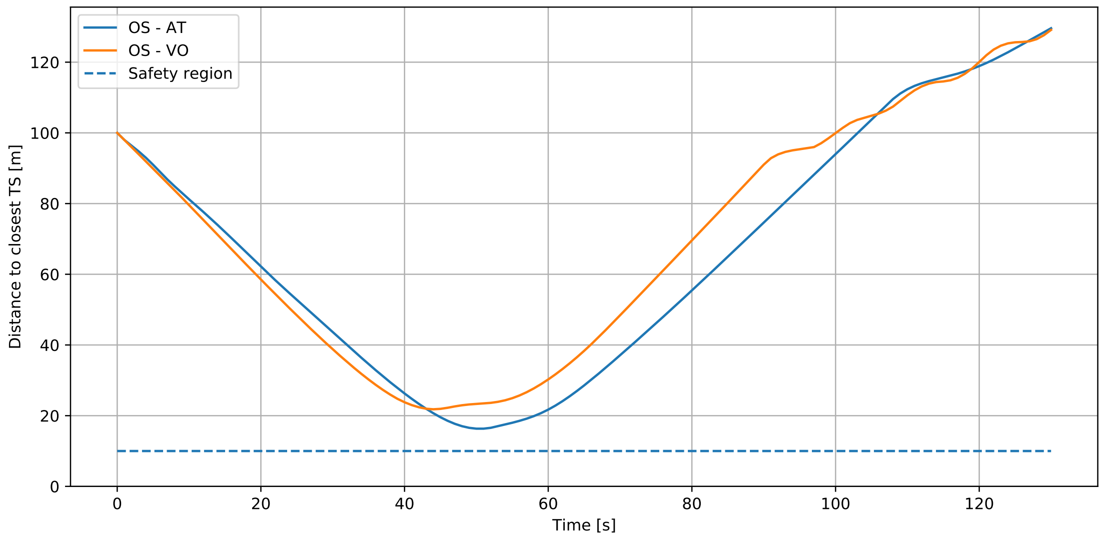


Figure 4.3: Situation 1: Head on. The distance from the OS trajectory to the closest TS for the different algorithms. Safety region marked with dashed line.

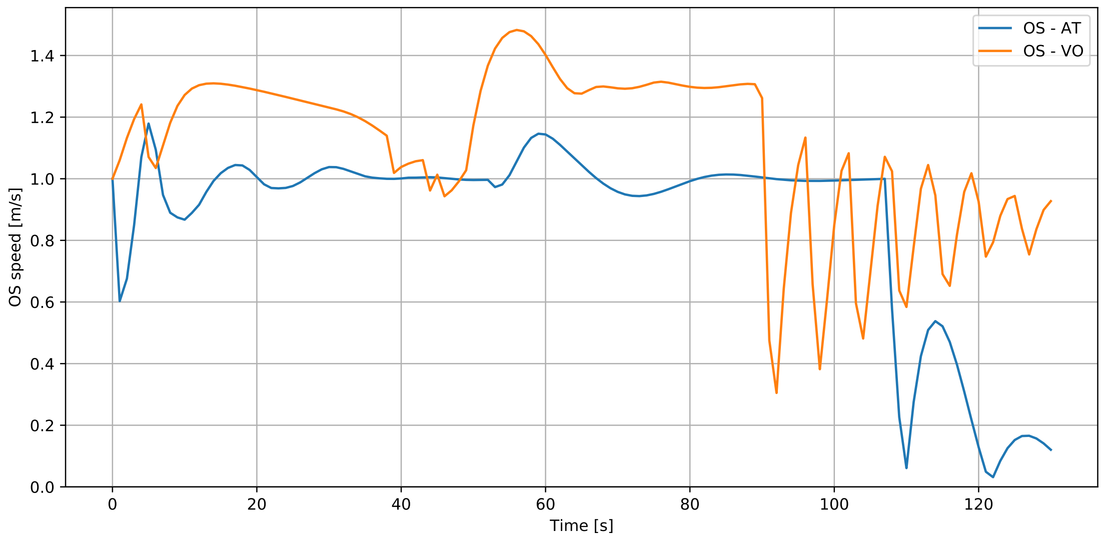


Figure 4.4: Situation 1: Head on. The speed of OS for the different algorithms.

4.3 Situation 2: Crossing a Channel With Traffic

In this situation the ASV must cross a channel, perpendicular to two vessels traversing it. In COLREG terms there are two different situations here. The encounter with the vessel coming from starboard is a "give way" encounter, while the encounter with the vessel coming from the port side is a "stand on" situation. The correct action is to give way to the vessel coming form the starboard side (the give-way encounter). The obligation to give way gets priority over *standing on* in the stand-on encounter.

SP-VP takes the simplest approach to this situation, as seen in Figure 4.5. It chooses to move slowly and wait until both crossing vessels has passed before moving towards the goal. This is a safe way to approach the situation, although somewhat defensive.

In the trajectory generated by VO on the other hand, the OS starts moving towards the goal immediately and does not slow down. Again, this is because VO does not have any concept of COLREG or the fact that this may create dangerous situations. This can be seen in Figure 4.6.

AT creates a trajectory in compliance with COLREG and passes in front of TS 2 and behind TS 1 as seen in Figure 4.7. The OS thus stands on in the stand on encounter and gives way in the give way encounter.

Figure 4.8 shows that the trajectory generated by AT comes closest to either of the TSs of the algorithms. This is caused by the way nodes are added to the graph during the graph building phase of the algorithm. When there is not much room, the nodes are added close to the obstacles, potentially at the border of the safety region. SP-VPs simple solution waiting for the TSs to pass creates the largest minimum distance to the TSs.

As for the speed profile, AT and VO are quite similar. They both keep the same speed throughout the scenario, before coming to a halt at the destination. SP-VPs speed profile is also similar, although delayed.

4.4 Situation 3: Narrow Strait With Meeting Traffic

In this situation the movement of the ASV is more constrained. An oncoming vessel is blocking a narrow strait and the ASV must wait until it is passed to go through the narrow strait.

SP-VP solves this in a simple way, as seen in Figure 4.10. It chooses to follow the nominal path slowly until the oncoming vessel is passed, before moving towards the goal at a higher speed. The trajectory chosen by VO does not work as well in this situation, as is evident from Figure 4.11.

The VO implementation has no concept of static obstacles, only dynamic obstacles. As such, it does not see the border around the working area and moves around the oncoming vessel, out of the working area. As such, the comparison with the other methods is not entirely fair. There are many variations of VO, including some that handle static obstacles.

The trajectory generated by AT is more reasonable. At the start, OS moves towards

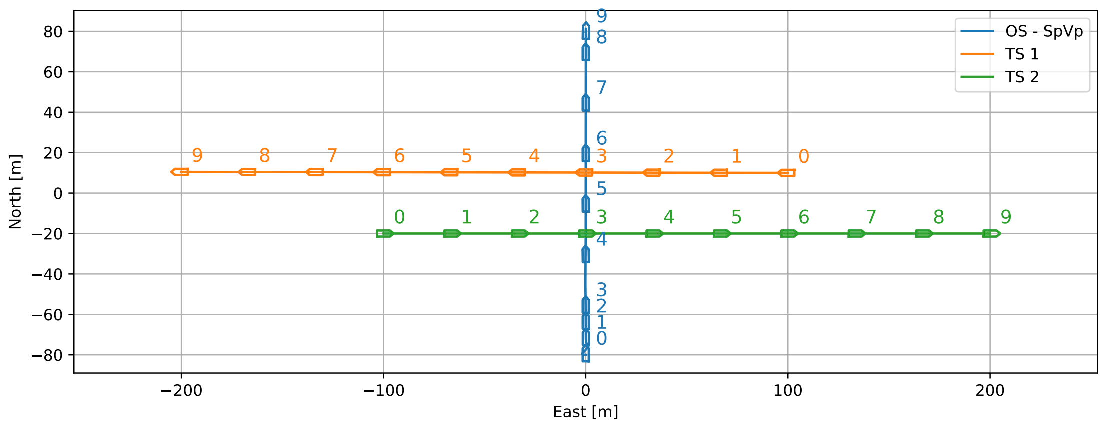


Figure 4.5: Situation 2: Crossing. The OS and two TSs in the xy plane. Trajectory generated by SP-VP. OS is travelling north and one TS travels from west to east while the other is travelling from east to west.

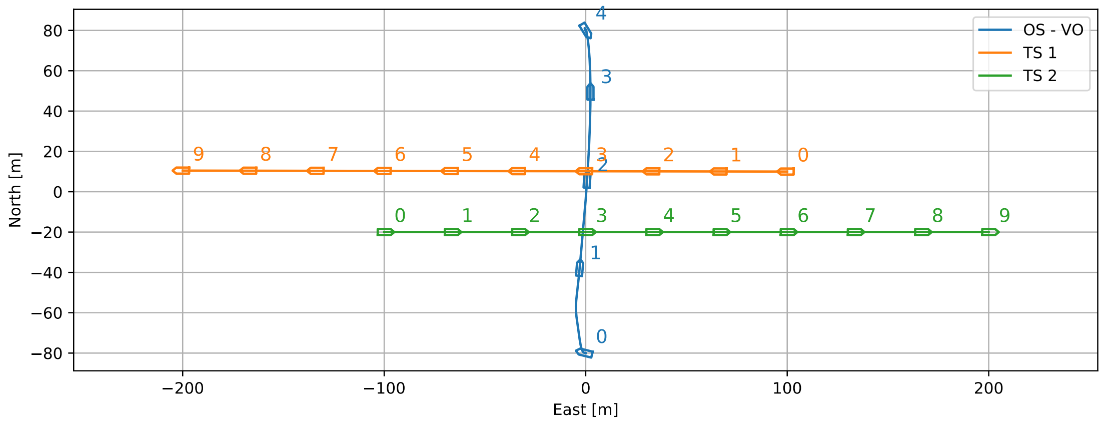


Figure 4.6: Situation 2: Crossing. The OS and two TSs in the xy plane. Trajectory generated by VO. OS is travelling north and one TS travels from west to east while the other is travelling from east to west.

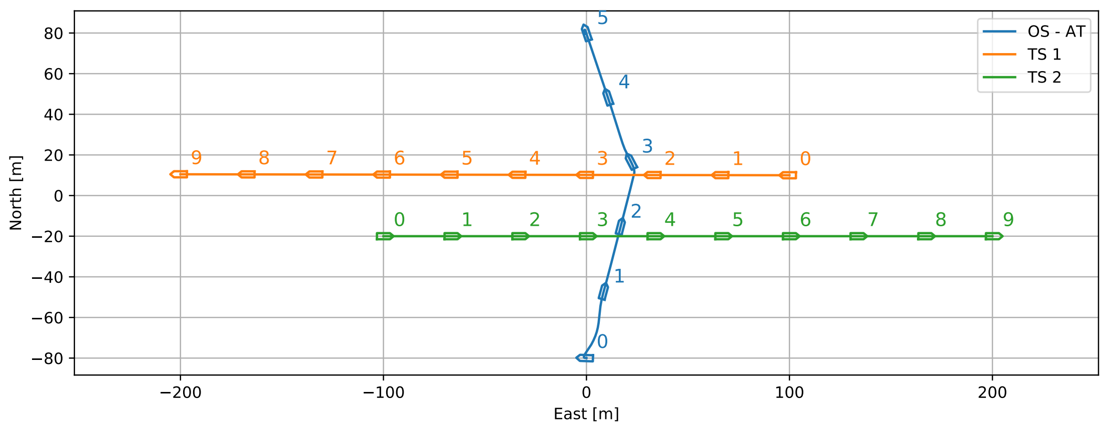


Figure 4.7: Situation 2: Crossing. The OS and two TSs in the xy plane. Trajectory generated by AT. OS is travelling north and one TS travels from west to east while the other is travelling from east to west.

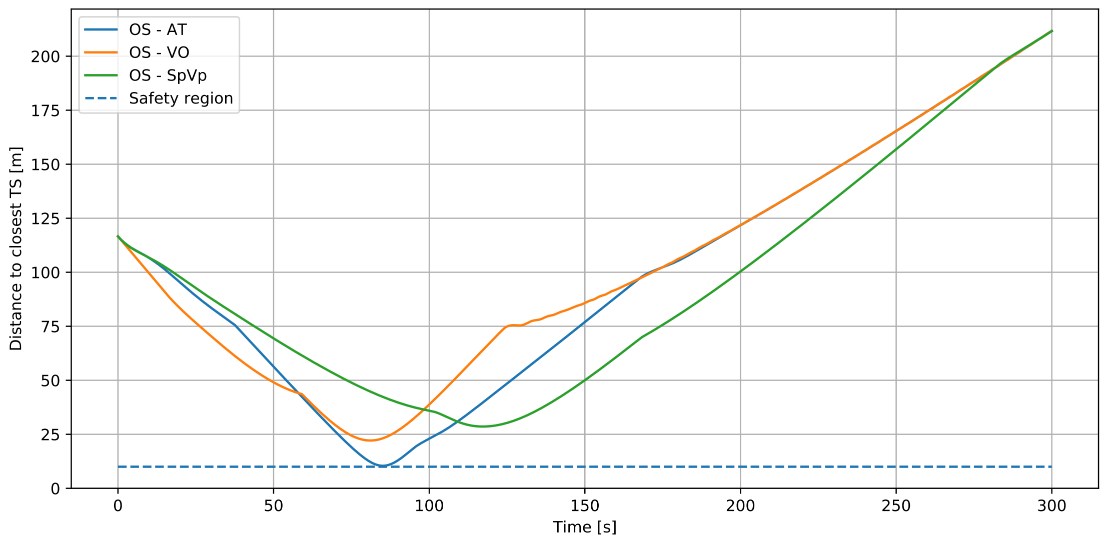


Figure 4.8: Situation 2: Crossing. The distance from the OS trajectory to the closest TS for the different algorithms.

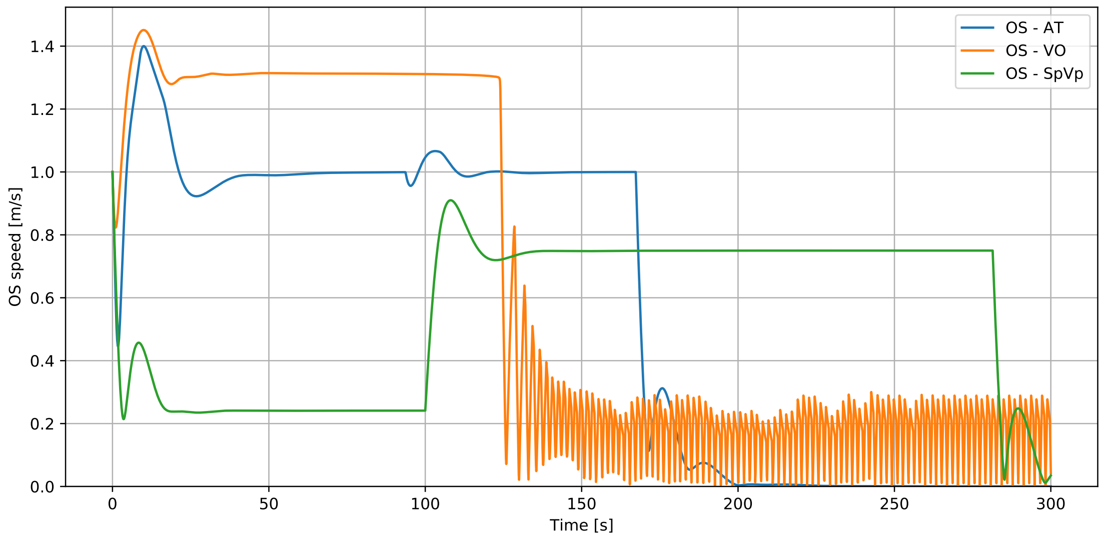


Figure 4.9: Situation 2: Crossing. The speed of OS for the different algorithms.

the opening of the strait. When it gets close, it moves behind the oncoming vessel which has passed. Finally, it follows the nominal path towards the destination. This is a reasonable trajectory. A safe distance to the TS is held while taking a shorter time to reach the destination than SP-VP, which has to wait for longer before moving towards the destination.

The same pattern as in the previous situation can be seen in the distance plot, Figure 4.13. AT creates a trajectory that touches the safety region, while the other to methods keep a somewhat larger distance. Once OS is past TS, the distance increases in a similar way for all the methods. In the speed plot, Figure 4.14, the turns can clearly be seen in for all the trajectories. For all the algorithms there is some oscillation when the heading is changed. For AT and SP-VP these are more visible because they follow a waypoint mission and the heading changes in potentially large steps.

4.5 Situation 4: Multiple Vessels Running The Proposed Method

Here, two different scenarios with multiple vessels running AT is presented. In the first one, shown in Figure 4.15, there are two vessels meeting in a head on encounter and another vessel travelling diagonally across the path of both of them. The second scenario, shown in Figure 4.16, there are two sets of head on situations, orthogonal to each other.

In the first scenario, the AT works reasonably well. The two vessels in the head on encounter pass each other port to port, compliant with COLREG. The third vessel, AT 3, crosses the trajectory of the two first ones and AT 1 gives way to it, in accordance with COLREG.

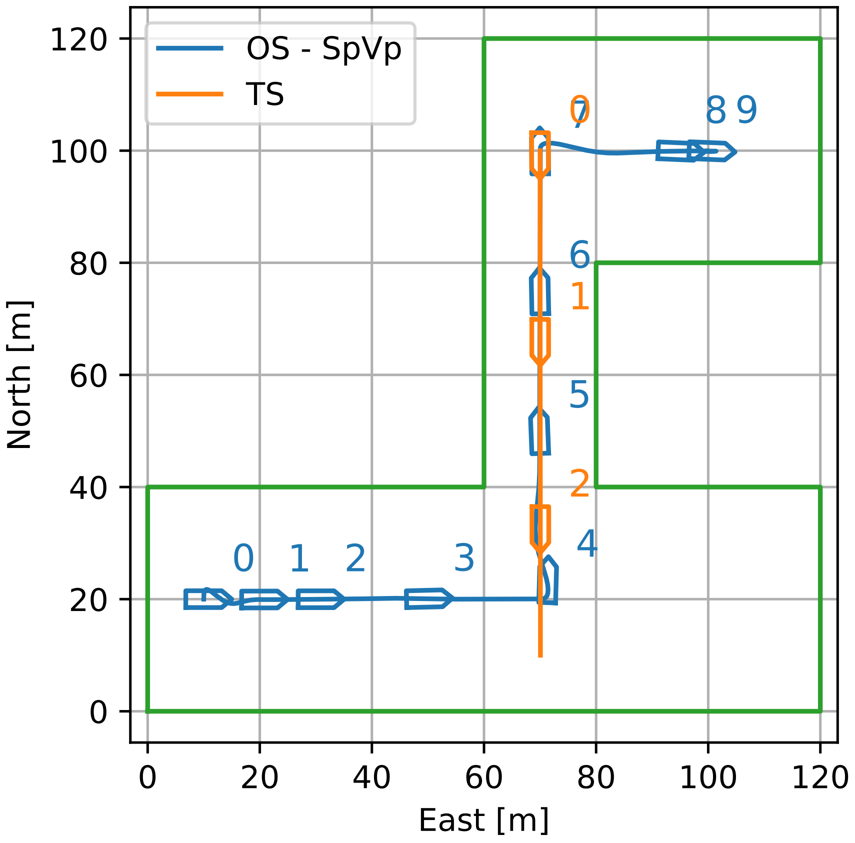


Figure 4.10: Situation 3: Narrow strait. OS travel towards the north-eastern corner of the working area. TS travels southwards and blocks the channel connecting the initial position of OS and its target destination. Trajectory generated by SP-VP. The border of the admissible area is marked in green.

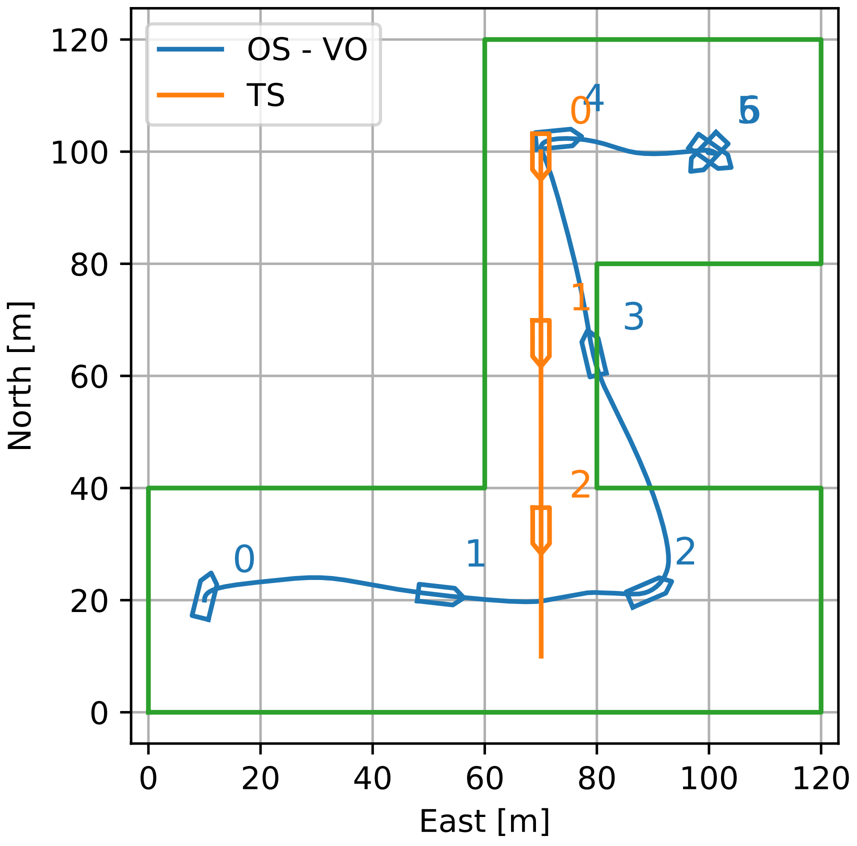


Figure 4.11: Situation 3: Narrow strait. OS travel towards the north-eastern corner of the working area. TS travels southwards and blocks the channel connecting the initial position of OS and its target destination. Trajectory generated by VO. The border of the admissible area is marked in green.

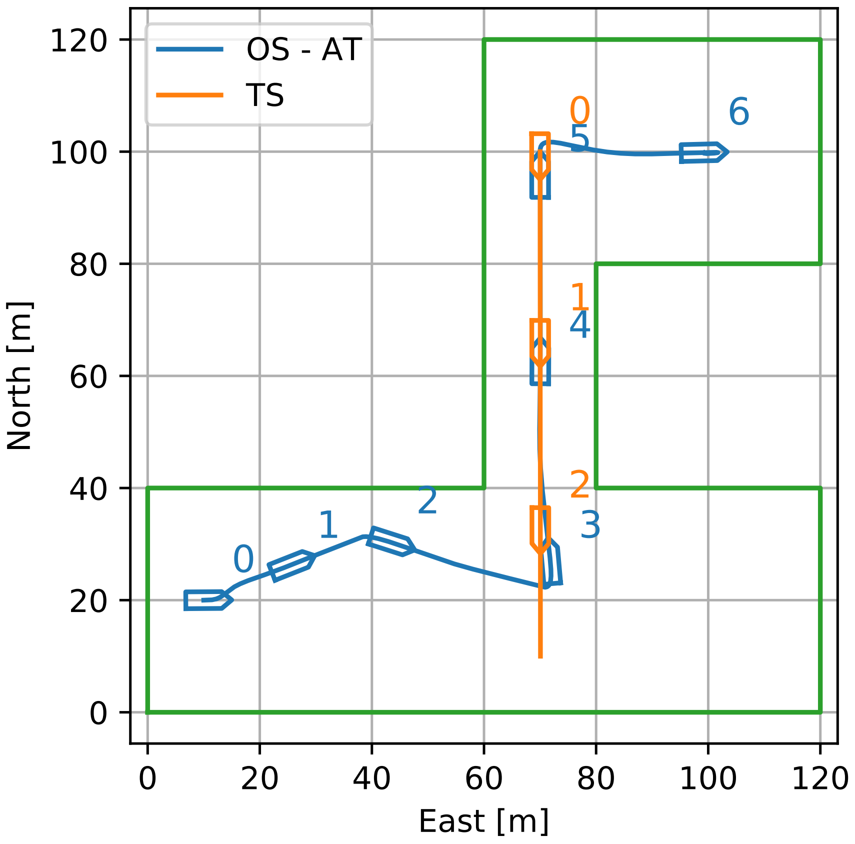


Figure 4.12: Situation 3: Narrow strait. OS travel towards the north-eastern corner of the working area. TS travels southwards and blocks the channel connecting the initial position of OS and its target destination. Trajectory generated by AT. The border of the admissible area is marked in green.

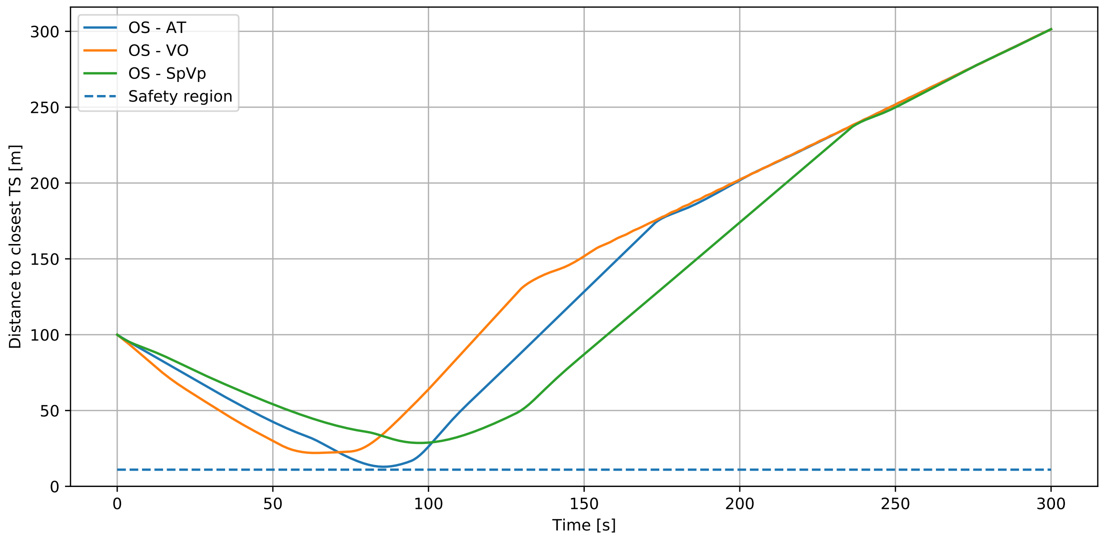


Figure 4.13: Situation 3: Narrow strait. The distance from the OS trajectory to the closest TS for the different algorithms.

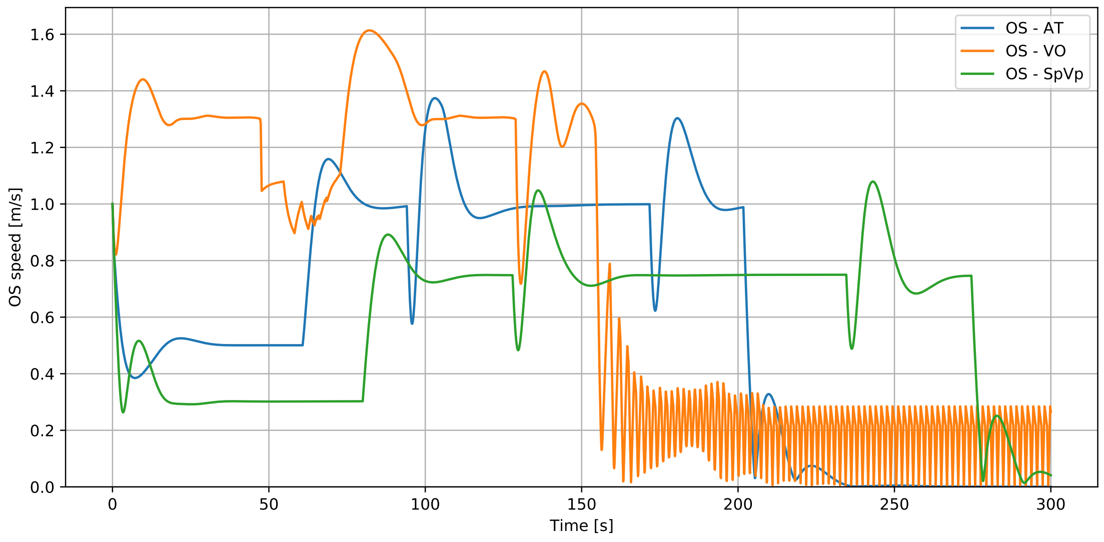


Figure 4.14: Situation 3: Narrow strait. The speed of OS for the different algorithms.

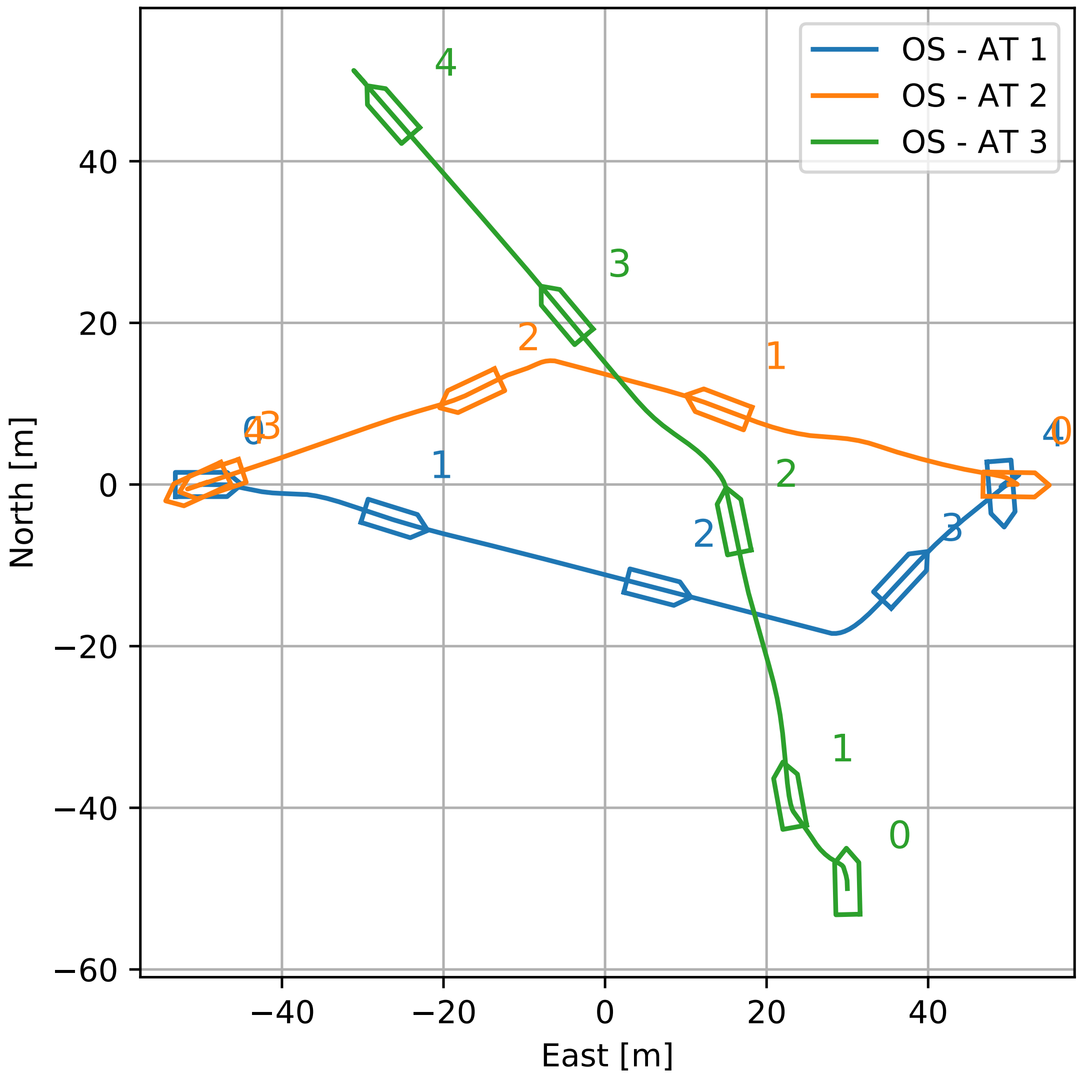


Figure 4.15: Situation 4: Multiple vessels running AT. Three vessels, all moving through the same area.

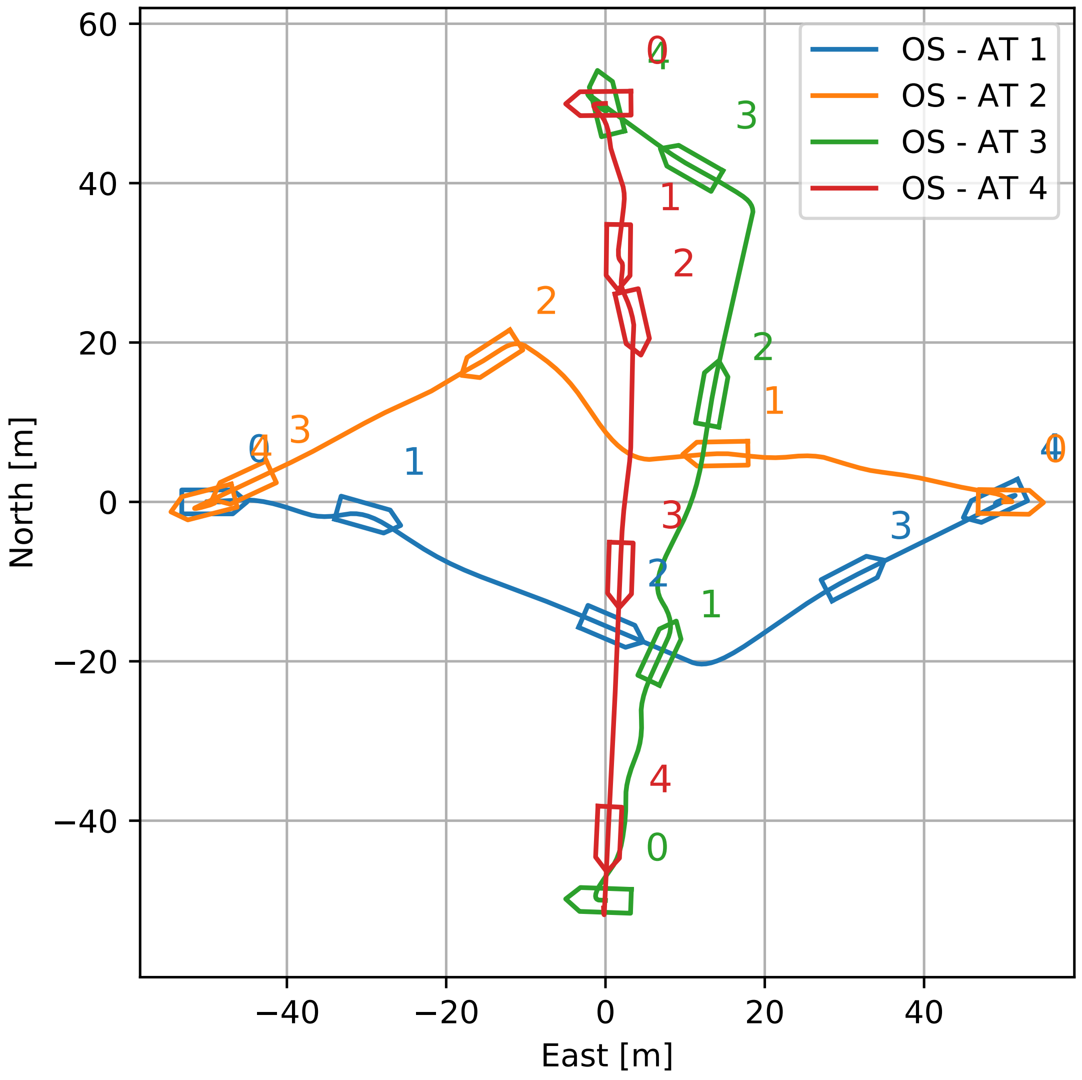


Figure 4.16: Situation 4: Multiple vessels running AT. Four vessels, all moving through the same area.

In the second scenario, the four vessels handles the situation less well. The trajectories are not as one would expect if a human controlled the vessel. This impacts the safety of the vessels quite severely. If a vessel does not make its maneuvers readily apparent to the other vessels, the risk of collision increases. This is especially true if there are humans in the system that expects that the other vessels maneuver as if they were controlled by experienced humans.

The reason for this odd behavior is the assumption that all dynamic obstacles move with constant speed and heading. When one of the vessels changes its speed or heading to avoid collision, this may lead to another vessel having to replan its trajectory. This replan may lead to a change in speed or heading which can further trigger replans in the other vessels. Thus there is a feedback loop here with replans triggering more replans. This effect is also seen during simulation of the system. In the scenarios where there are multiple vessels making decisions and thus not following a constant speed and heading, there are more replans than in the simple situations where that assumption holds.

A possible solution to this is for the vessels to cooperate and share their planned trajectories with each other. If the trajectories of the other vessels are known during the first planning, there should be no need to replan later, given that the other vessels follow the trajectory they shared.

4.6 Situation 5: Crowded Area

In the final situation, the possible maneuvers of OS is limited by multiple TSs moving with constant speed and heading in different directions. As seen in Figure 4.17, the trajectory does not follow all the rules from COLREG. For example does OS not give way for TS 3, even though it is in a give way situation. The TS is however quite a distance away when the heading change is done and when OS crosses the trajectory, so the violation is not too severe. In Figure 4.18 however, it can be seen that the distance to the other vessels becomes quite small halfway to the destination. During the turn close to the crossing of the paths of TS 1 and TS 5, the distance to the TSs goes slightly into the safety region. When turning, OS does not stick to the planned trajectory exactly. By design, the safety region includes this type of deviations from the trajectory. The breach of the safety region is thus unlikely to lead to a collision.

4.7 Discussion

In all the situations seen, the proposed method was able to find a collision free trajectory, from the start position to the destination. In most cases the chosen trajectory is also reasonable and follows COLREG. However, there are situations where the performance of the proposed method is suboptimal. This is especially apparent in Figure 4.16 where Assumption 3.4 regarding constant velocity and heading for obstacles is does not hold. When this assumption is broken, the planned trajectory may not work and lead to a collision, forcing the algorithm to replan to generate a new trajectory. When a replan happens, the trajectory may be changed significantly, resulting in maneuvers that differ

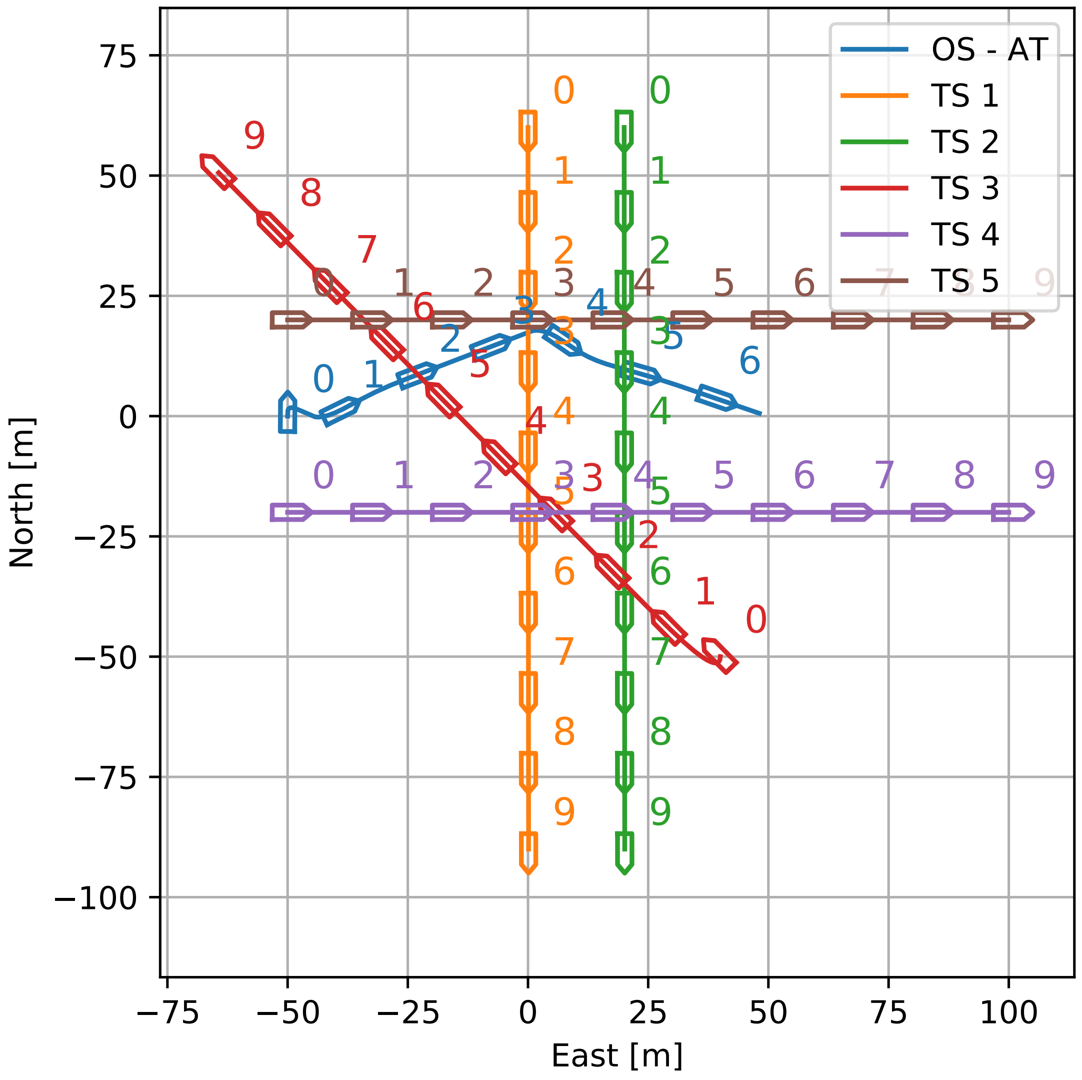


Figure 4.17: Situation 5: Crowded area. Multiple vessels moving parallel and orthogonal to the OS, restricting the maneuvers available.

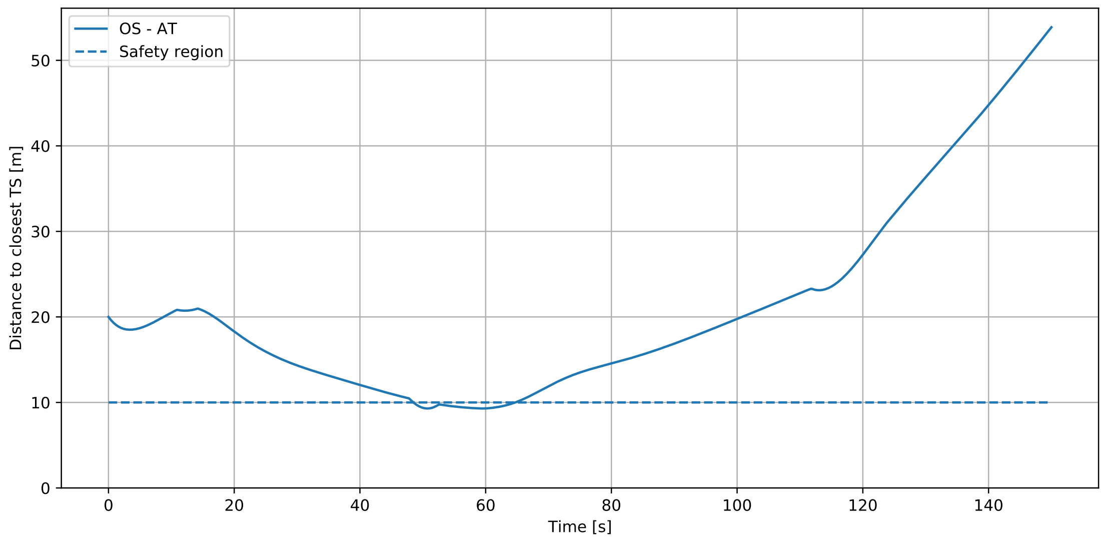


Figure 4.18: Situation 5: Crowded area. The distance from the OS trajectory to the closest TS.

from the assumptions from other vessels in the area. Unexpected maneuvers decrease the safety of the system. Other seafarers expect trajectory changes to be made early, in accordance with COLREG. Late trajectory changes thus lead to increased risk of collision.

Another way Assumption 3.4 can be broken is through vessel mechanics. Even though a vessel has a constant speed and heading reference, it may not track it perfectly. Measurement noise or estimation errors in the pose of the TSs may also lead the current trajectory of the OS to become infeasible if it is assumed that the TSs continue with their current speed and heading. One could possibly implement a better method for estimating the future speed and heading of the TSs. Such an implementation is outside the scope of this thesis and is left for future work. Two features that are implemented to combat this are the scaling of dynamic obstacles, and the limited time horizon. Firstly, the representation of the dynamic obstacles is scaled by with the distance from the OS so that nodes added to the graph are further from the obstacles. Small changes in speed or heading will thus impact the trajectory less, giving increased safety margin. Secondly, the trajectory of the TSs is only projected for a certain time into the future. Because the estimate for pose of the TSs far into the future is uncertain, especially in confined waters, AT does not use this to create the graph. It was found during development of the method that the generated trajectories became simpler when it did not rely on uncertain TS poses far into the future.

Sudden changes in speed or heading, which follows from the waypoint mission the OS follows, may decrease passenger comfort. A way to combat large changes in speed or heading is to add a term for this in the cost function. However, this should be weighted much less than the terms contributing to the safety of the system. Any such change to the cost function must not make the algorithm choose a significantly less safe trajectory to increase passenger comfort. AT does better than VO and SP-VP in some regard here. VO changes both speed and heading arbitrarily to avoid collision. SP-VP is unable to do heading changing maneuvers, so must change the speed along the nominal path fairly often. AT can however change both the speed and the heading of OS. In that way, the changes in each of speed and heading be lower. AT may for example do a small course changing maneuver to avoid an obstacle, whereas SP-VP may need to stop completely to wait for the obstacle to pass.

COLREG defines the classification of encounters in terms of two vessels. When there are multiple vessels all in the same area the classification is not defined directly. The approach used in this thesis is to classify the encounter between the OS and each TS separately. This is a simple approach which seems to work, at least in simple situations. However, it does not consider the interaction between the different TSs. It may not be reasonable for a given TS to follow the maneuvers expected from the COLREG classification of the encounter with OS because it has obligations against some other TS. There is ongoing research into creating COLREG-compliant algorithms for multi-vessel encounters, for example [Zhao and Roh (2019)] and [Cho et al. (2020)].

Chapter 5

Conclusion and Future Work

This chapter summarizes and concludes the work done in this thesis and suggests items for future work.

5.1 Conclusion

This thesis presents the a method for planning collision-free trajectories for an Autonomous Surface Vehicle (ASV) operating in confined waters in the presence of other vessels. The method is graph-based, where firstly the relevant features of the operational environment is represented in an area-time space, and subsequently, a graph i built and searched to find the optimal trajectory through the space. The method is tested several simulations, and compared to two other method for maritime Collision Avoidance (COLAV), namely the Single Path Velocity Planner (SP-VP) and Velocity Obstacle (VO) algorithms.

In all the situations seen in Chapter 4, Area-Time trajectory planning method (AT) was able to find a collision free trajectory from the start position to the desired destination that keeps within the assigned operating area. By virtue of the way the graph is built, each leg of the generated waypoint mission is also feasible given the vessel characteristics. However, acceleration and yaw rate constraints are not taken into account. This may lead to increased tracking errors or decreased passenger comfort.

In many of the cases, AT chooses the correct behavior according to The International Regulations for Preventing Collisions at Sea (COLREG). This holds especially true in the simple situations where the assumptions about each obstacle keeping a constant speed and heading holds. When the obstacles break this assumption the performance of the algorithm deteriorates. Because the information the trajectory was generated on was false it may lead to collisions and a replan is necessary. Frequent replans lead to trajectories that are unpredictable from the viewpoint of other vessels, which decreases safety.

AT improves on SP-VP by giving more freedom in the maneuvers available. For example, in a head on encounter, AT can change the course of Ownship (OS), while SP-VP is unable to move OS away from the preplanned path and the problem becomes infeasible. By being able to choose course changing maneuvers the trajectories become more natural and complies better with COLREG, which prefers course alteration over speed alteration. However, this improvement comes at a cost. By adding another dimension to the problem the implementation becomes more complex. There are more geometrical special cases which has to be taken care of and the search space becomes larger. With the increased complexity there is also potential for increased runtime.

The runtime of the algorithm is heavily dependent on the size of the graph that is being built. By increasing the size of the sets of feasible speeds and waiting intervals, the graph better captures the possible trajectories, however this comes at the cost of increased graph build time. Another consideration related to runtime is the size of the heuristic used during building the graph in comparison with the size of the cost function. The cost function has no bearing for which direction the goal position is in. The heuristic is what drives the graph building towards the goal. If the heuristic is too low, the graph building step may take a long time to find a path to the goal position. However if the cost function is not weighted high enough, the generated graph will not incorporate the correct COLREG behavior.

5.2 Future Work

The following items are suggested for future work:

- Extending the AT method to be cooperative and take advantage of knowing the trajectory of other vessels. This can give additional safety because a larger system can behave in an expected way.
- Estimating the trajectory of obstacles and taking advantage of this information. This can lead to trajectories that require fewer replans.
- Extending the method to take acceleration and yaw rate constraints into account. This can create more realistic trajectories that are guaranteed to be feasible for the vessel. It may also take passenger comfort into account.

Bibliography

[Bellamy-Royds et al. (2020)] Bellamy-Royds, A., Bah, T., Lilley, C., Schulze, D., Williigers, E., 2020. Scalable vector graphics (svg) 2 w3c editor's draft URL: `https://svgwg.org/svg2-draft/painting.html#FillRuleProperty`.

[Cho et al. (2020)] Cho, Y., Han, J., Kim, J., 2020. Efficient colreg-compliant collision avoidance in multi-ship encounter situations. IEEE Transactions on Intelligent Transportation Systems , 1–13doi:`10.1109/TITS.2020.3029279`.

[Dijkstra (1959)] Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271. doi:`10.1007/BF01386390`.

[Fiorini and Shiller (1998)] Fiorini, P., Shiller, Z., 1998. Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research 17, 760–772.

[Guo and Zavlanos (2018)] Guo, M., Zavlanos, M.M., 2018. Multirobot data gathering under buffer constraints and intermittent communication. IEEE Transactions on Robotics 34, 1082–1097. doi:`10.1109/TRO.2018.2830370`.

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

[International Maritime Organization (1972)] International Maritime Organization, 1972. Convention on the international regulations for preventing collisions at sea, 1972 (COLREGs) .

[Korf (2000)] Korf, R.E., 2000. Recent progress in the design and analysis of admissible heuristic functions, in: Choueiry, Berthe Y.and Walsh, T. (Ed.), Abstraction, Reformulation, and Approximation, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 45–55.

[Lozano-Pérez and Wesley (1979)] Lozano-Pérez, T., Wesley, M.A., 1979. An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM 22, 560–570. doi:`10.1145/359156.359164`.

[Pedersen (2019)] Pedersen, A.A., 2019. Optimization Based System Identification for the milliAmpere Ferry. Master's thesis. Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

[Tam and Bucknall (2010)] Tam, C., Bucknall, R., 2010. Collision risk assessment for ships. Journal of Marine Science and Technology 15, 257–270. doi:10.1007/s00773-010-0089-7.

[Thyri et al. (2020)] Thyri, E.H., Breivik, M., Lekkas, A.M., 2020. A path-velocity decomposition approach to collision avoidance for autonomous passenger ferries in confined waters, in: IFAC World Congress, Berlin, Germany.

[Zhao and Roh (2019)] Zhao, L., Roh, M.I., 2019. COLREGs-compliant multiship collision avoidance based on deep reinforcement learning. Ocean Engineering 191, 106436. URL: https://www.sciencedirect.com/science/article/pii/S0029801819305840, doi:https://doi.org/10.1016/j.oceaneng.2019.106436.