Hybrid Collision Avoidance for Autonomous Surface Vessels

Einvald Serigstad

Master of Science in Cybernetics and Robotics

Submission date:	June 2017
Supervisor:	Morten Breivik, ITK
Co-supervisor:	Bjørn-Olav H. Eriksen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

Further development of the Dynamic Window (DW) algorithm for ASVs is the main goal of this master thesis. In particular, a problem escaping the avoidance region if entered should be investigated.

The DW algorithm should be included in a hybrid COLAV method. A common deliberate method should be used in the hybrid COLAV method, where an interface between the reactive and deliberate method must be developed.

Any modifications to the DW algorithm proposed in the master thesis should be compared to the DW algorithm with ASVs.

- Further develop the DW algorithm and enable it to exit the avoidance region once entered. However, the algorithm should perform at the same level or better when not inside the avoidance region, compared to the modified DW algorithm.
- Let the ASV trajectory prediction be defined by multiple (2 or more) velocity pairs. How does this affect the computational load and performance?
- Introduce a deliberate method and an interface between the deliberate method and DW algorithm to form a hybrid COLAV method.
- Introduce performance metrics to measure the performance level when simulating and comparing the different algorithms.

Preface

This thesis is written as a compulsory part of the Master's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). The work on this thesis, carried out during the spring semester 2017, have resulted in an increasing interest and understanding for the field of collision avoidance.

I want to express my gratitude towards my supervisor Morten Breivik and co-supervisor Bjørn-Olav H. Eriksen for the feedback and enthusiasm provided during the semester.

Einvald Serigstad

Trondheim, June 2017

Abstract

When developing autonomous vehicles, collision avoidance (COLAV) methods are essential. COLAV methods are generally divided into two groups; reactive and deliberate methods. Reactive COLAV methods act solely on sensor data from the immediate environment, while deliberate methods generate global paths based on available data of the complete environment. The focus of this thesis is mainly on a reactive COLAV method named Dynamic Window (DW) algorithm. The DW algorithm has previously been modified to consider vehicles with second order nonholonomic constraints, such as autonomous surface vehicles (ASVs). This modification is denoted Algorithm A and is further developed in this thesis. Algorithm A introduces an avoidance region around obstacles, which the vehicle is motivated to avoid. If the vehicle is inside the avoidance region, however, Algorithm A will not be motivated to leave the region and will lose the ability to evaluate the vehicle distance towards obstacles. Furthermore, the region may cause problems when travelling through narrow passages.

Two new DW algorithms, Algorithm B and Algorithm C, are introduced in this thesis to cope with the avoidance region issues. While Algorithm A focuses purely on reaching the goal without entering the avoidance region, Algorithm B will also try to exit the region if entered. Algorithm C is less hesitant to enter the avoidance region but prefers trajectories that has a minimum portion inside the avoidance region.

To avoid being trapped in a local minima and enable the use of predefined trajectories, a hybrid COLAV method consisting of a reactive and a deliberate method is introduced. The DW algorithm is modified to suit the hybrid method, resulting in the Hybrid Dynamic Window (HDW) algorithm. The algorithm works both as a reactive COLAV method, and an interface between the reactive and deliberate method. The Rapidly-Exploring Random Tree (RRT) algorithm, which is a deliberate method, generates a planned trajectory towards the goal for the HDW algorithm to track. The HDW algorithm tracks the planned trajectory by inputting a desired ASV trajectory that aligns as good as possible with the planned trajectory.

A simulation environment and an ASV model are implemented to assess the new algorithms based on several performance metrics. The algorithms are tested in various scenarios designed to challenge potential weaknesses of the algorithms. A new performance metric, integral of distance inside avoidance region (IDI), is introduced to measure the algorithm performance.

The introduction of Algorithm C greatly improves the performance when operating close to or inside the avoidance region. Furthermore, applying the modifications introduced in Algorithm C to the HDW algorithm guided by the RRT algorithm, results in a consistently good ASV behaviour, where the ASV always keeps a safe distance to obstacles.

Sammendrag

Når man utvikler autonome overflatekjøretøy (ASVer) er gode metoder for kollisjonsunngåelse (COLAV-metoder) essensielle. COLAV-metoder deles generelt inn i reaktive og globale metoder. Reaktive metoder handlers basert på sensordata fra de nærmeste omgivelsene. Globale metoder bruker all tilgjengelige informasjon om hele omgivelsen for å planlegge en global bane til målet. Hovedfokuset i denne masteroppgaven er en reaktiv algoritme, Dynamic Window-algoritmen (DW). DW-algoritmen er tidligere tilpasset kjøretøy med andre ordens ikke-holonomiske beskrankninger, som for eksempel ASVer. Den modifiserte algoritmen vil bli videreutviklet i denne masteroppgaven og er omtalt som Algoritme A. For å motivere kjøretøy til å holde en viss avstand fra hindringer, introduserer Algoritme A unngåelsesområder rundt hindringene. Unngåelsesområdet skaper problemer når kjøretøyet er inne i unngåelsesområdet da Algoritme A ikke har motivasjon for å lede kjøretøyet ut området. I tillegg mister algoritmen evnen til å vurdere avstand til hindringer. Unngåelsesområdet kan også skape problemer om man ønsker å kjøre gjennom smale passasjer.

Denne masteroppgaven introduserer to nye DW algoritmer, algoritmene B og C, for å håndtere utfordringene rundt unngåelsesområdet. Algoritmene A og B er identiske når kjøretøyet er utenfor unngåelsesområdet. Hvis kjøretøyet er inne i området, derimot, vil Algoritme B være motivert til å velge baner som leder ut av området. Algoritme C velger bane basert på hvor stor del av banen som er inne i unngåelselsområdet og er derfor mindre påvirket av om kjøretøyet er innenfor ellet utenfor området. Alle algoritmene ønsker å velge baner som leder mot målet.

En hybrid COLAV-metode bestående av en reaktiv og en global metode er introdusert for å unngå å bli fanget i et lokalt minima. DW-algoritmen er tilpasset den hybride metoden, noe som resulterer i en Hybrid Dynamic Window-algoritme (HDW) som opptrer både som en reaktiv COLAV-metode og et grensesnitt mellom den reaktive og den globale metoden. En global metode, Rapid-Exploring Random Tree-algoritme (RRT), genererer en planlagt bane til målet som HDW-algoritmen sporer ved å velge baner som samstemmer mest mulig med den planlagte ASV-banen.

Et simuleringsmiljø er implementert med en ASV-modell for å vurdere de nye algoritmene i flere scenarioer, hvor vurderingen er basert på flere prestasjonsmetrikker. En ny presatsjonsmetrikk, integralet av avstanden inne i unngåelsesområdet (IDI), er introdusert for å måle prestasjonene til algoritmene.

Å bruke modifikasjonen som er blitt introdusert i Algoritme C i HDW-algoritmen og la den bli guidet av den planlagte RRT-banen resulterer i en hybrid COLAV-metode som gir en jevnt god ASV-oppførsel, hvor ASVen alltid holder en relativt trygg avstand til nærliggende hindringer.

Nomenclature

$\boldsymbol{\eta}$	ASV pose
$\boldsymbol{\nu}$	ASV velocity
$\boldsymbol{M}$	Mass matrix
$\boldsymbol{C}(\boldsymbol{\nu})$	Coriolis and centripetal matrix
$\boldsymbol{D}(\boldsymbol{\nu})$	Damping matrix
$\boldsymbol{\tau}$	Force vector
$\boldsymbol{B}$	Input matrix
$\boldsymbol{f}$	Throttle vector
u	Surge speed
v	Sway speed
r	Yaw rate
X	Surge force
Y	Sway force
N	Yaw moment
l_r	Rudder arm length
δ_ψ	Rudder angle
K_{δ_ψ}	Rudder coefficient
Δ	Lookahead distance
$\bar{\mathcal{C}}_{free}$	Free configuration space
$\bar{\mathcal{C}}_{forb}$	Forbidden configuration space
T_s	DW period
$\boldsymbol{\Omega}$	Avoidance region
$\boldsymbol{\mathcal{T}}$	Antitarget region
α	Yaw rate function weight
β	Distance function weight
γ	Velocity function weight
$\bar{\alpha}$	HDW align and distance function weight
c	Distance function B weight
κ	Distance function C weight
ϵ	RRT bias towards goal
$r_{\boldsymbol{\Omega}}$	Avoidance region radius
$r_{\boldsymbol{\mathcal{T}}}$	Antitarget region radius
T	Simulation runtime
IADC	Integral of absolute differentiated control
IAE	Integral of absolute error
IDI	Integral of distance inside avoidance region
$D_{\boldsymbol{\mathcal{T}}}$	Distance to avoidance region
W	Energy consumption

Abbreviations

ASV	Autonomous Surface Vehicle
AUV	Autonomous Underwater Vehicle
COLREGS	International Regulations for Avoiding Collisions at Sea
DOF	Degrees Of Freedom
DW	Dynamic Window
HDW	Hybrid Dynamic Window
LOS	Line Of Sight
MPC	Model Predictive Control
NED	North East Down
RRT	Rapidly-Exploring Random Tree
TA	Trajectory Alignment
UGAS	Uniformly Globally Asymptotically Stable
ULES	Uniformly Locally Exponentially Stable
VO	Velocity Obstacles

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the topic and structure of the thesis. In addition, a review of previous works and contributions of this thesis are presented.

1.1 Background and Motivation

Autonomous surface vessels (ASVs) require a collision avoidance (COLAV) method when moving through environments with obstacles. In unknown dynamic environments, the ASV needs a real-time COLAV method that receives sensor data of the surrounding environment [9]. However, a reactive method which acts solely on sensor data is only able to consider the immediate environment, hence the ASV needs a deliberate path planner to be guaranteed to reach the goal.

Numerous reactive and deliberate COLAV methods exist for vehicles with holonomic constraints and first order nonholonomic constraints. Few results are, however, presented where marine vessels with second order nonholonomic constraints are considered [12]. Holonomicity are further described in Section 2.1. To employ autonomous marine vessels, a reliable COLAV method is needed. This motivates a further development of COLAV methods for marine vessels such as ASVs. Figure 1.1 C-worker shows the C-Worker ASV, which is an ASV applied for monitoring in oil and gas operations.

The Dynamic Window (DW) approach [17] is a reactive COLAV method intended for robots with first order nonholonomic constraints. The method is adapted and tested for ASVs in [28], where it is concluded superior compared to several other reactive COLAV methods, e.g. the potential field method [23, 24]. Further, the DW algorithm is modified in [12, 13] to account for the second order nonholonomic constraints and time-varying acceleration limits of AUVs. This modification is simulated for ASVs and compared to the original DW algorithm in [32], which this thesis is a continuation of. The horizontal modelling and control of AUVs are similar to ASV modelling, hence the modified DW

Figure 1.1: The C-Worker is a multiuse offshore ASV developed to conduct subsea positioning, surveying and environmental monitoring for oil and gas operations. Courtesy of [1].

algorithm is applicable to ASVs as well. The modifications to the DW algorithm presented in [12, 13] greatly improves the COLAV performance of the DW algorithm used for horizontal COLAV on AUVs. To penalize trajectories that are close to obstacles, an avoidance region is added around obstacles. The avoidance region improves the DW algorithm when the vehicle is outside the region. When the vehicle is inside the region, on the other hand, the chance of colliding is greatly increased, compared to the original DW algorithm [32]. Furthermore, the algorithm is a reactive COLAV method and can not guarantee that it guides the ASV to the goal. This motivates the development of a hybrid COLAV method which utilizes both the DW algorithm and a deliberate COLAV method. Hence, the main focus of this thesis is to modify the DW algorithm to account for the weaknesses discovered in [32] and to develop an interface between the DW algorithm and a deliberate method for use in a hybrid COLAV method.

1.2 Previous Work

Numerous reactive and deliberate COLAV methods are developed over the years. Reactive methods consider only the immediate environment based on available sensor data, while deliberate methods generate global path based on stored environmental data. The Potential field method [23, 24], Vector Field Histogram [5], Velocity Obstacles (VO) [15] and the Dynamic Window (DW) [17] algorithm are all examples of popular reactive COLAV methods, while Rapidly-Exploring Random Trees (RRT) [27], Voronoi Diagram [11], Probabilistic Roadmap (PRM) [22] and A* algorithm [19] are known deliberate methods. The A*, RRT and PRM algorithms are compared in [40], where it is concluded that the RRT algorithm is fast and generates smooth paths, while A* and PRM have are expected to have higher computation cost and return shorter paths.

The DW algorithm is further developed and generalized for use on a greater variety of vehicles in [9]. Modifications to improve the DW algorithm for use on marine vessels are presented in [28]. Further, [12, 13] presents a modification to the DW algorithm which takes the second order nonholonomic constraints of an AUV into account and introduces the avoidance region around the obstacles. This modified algorithm is further reviewed and compared to the original DW algorithm in [32], where suggestions for further developments are presented. A further review of the DW algorithm is given in Section 2.4.1.

Several hybrid COLAV methods applying the DW algorithm are introduced. In [29], the DW algorithm is used as a model predictive control (MPC) method using a control Lyapunov function (CLF), which yields convergence to the goal position. The DW algorithm is used with a modified RRT algorithm, where the A* algorithm operates as an RRT guide in [28], and tested for ASVs. The RRT algorithm is further reviewed in Section 2.5.1 and is used in a hybrid COLAV method in this thesis. Furthermore, the hybrid COLAV method concept is described in Section 2.6

For marine vessels, COLAV methods taking the International Regulations for Avoiding Collisions at Sea (COLREGS) into account are proposed. A hybrid COLAV method using the DW algorithm and is compatible with COLREGS is presented in [28]. Furthermore, the VO method has been modified to take COLREGS into account and scales well with trafficked environments [25].

1.3 Contributions

The contributions of this report are:

- A review of the DW algorithm and existing extensions.
- A further development of the modified DW algorithm in [12, 13], compensating for algorithm weaknesses addressed in [32]. The weaknesses mainly concern the avoidance region blinding the DW distance function if the ASV is close to or inside the region. Two algorithms, algorithms B and C, which takes this into account are proposed.
- A new Hybrid Dynamic Window (HDW) algorithm enabling the DW algorithm to function as both a reactive method and a trajectory tracker in

a hybrid COLAV system. The HDW algorithm simplifies the architecture and tuning of the system and utilizes the planned trajectory in near future to assess how well the predicted ASV trajectories align with it.

- Extensive testing through simulations of the new reactive DW algorithms and the HDW algorithm in a hybrid COLAV method with the RRT algorithm as a deliberate method.
- A new performance metric, integral of distance inside avoidance region (IDI). The IDI metric offers information of how far and long the vehicle resides inside the avoidance region.

1.4 Outline of the Report

This thesis is divided as follows: Chapter 2 provides a theoretical background for vessel modelling and control, guidance systems, and collision avoidance methods focusing especially on the DW algorithm and RRT algorithm. Three modified versions of the DW algorithm are presented in Chapter 3. Chapter 4 introduces the HDW algorithm which functions both as a reactive method and a trajectory tracker, intended for use in a hybrid COLAV method. Chapter 5 describes the implementation of a simulator used to test the reactive DW algorithms and the hybrid COLAV method presented in chapters 3 and 4, respectively. Chapter 6 describes the different simulation scenarios and the simulation results for both the reactive and the hybrid COLAV method. Finally, Chapter 7 concludes the thesis and proposes further work.

Chapter 2

Theoretical Background

Theoretical background on marine vessel modelling, guidance systems, spaces in motion planning and COLAV methods are necessary to fully understand the details of the thesis and the presented results. A brief review is given in this section.

2.1 Vessel Modelling

The marine craft equations of motion can be written in a vectorial setting as [16]

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\Theta}(\boldsymbol{\eta})\boldsymbol{\nu}, \tag{2.1}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{g}_0 = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}}. \tag{2.2}$$

The matrices $\boldsymbol{M}$, $\boldsymbol{C}(\boldsymbol{\nu})$ and $\boldsymbol{D}(\boldsymbol{\nu})$ denotes the vehicle inertia, Coriolis and centripetal, and damping matrices, respectively. Generalized gravitational and buoyancy are given by $\boldsymbol{g}(\boldsymbol{\eta})$ and forces and moments due to ballast and water tanks are given in $\boldsymbol{g}_0$. The vectors

$$\boldsymbol{\eta} = [x, y, z, \phi, \theta, \psi]^{\top} \in \mathbb{R}^3 \times SO(3) \tag{2.3}$$

and

$$\boldsymbol{\nu} = [u, v, w, p, q, r]^{\top} \in \mathbb{R}^6 \tag{2.4}$$

describe the vessel pose and velocity, respectively, using the SNAME(1950) notation for marine vessels. The velocity $\boldsymbol{\nu}$ is further described in Section 2.1.1 and illustrated in Figure 2.1. The pose $\boldsymbol{\eta}$ is further described in sections 2.1.1 and 2.1.2. The actuator forces and moments are given by $\boldsymbol{\tau} \in \mathbb{R}^6$, while $\boldsymbol{\tau}_{\text{wind}} \in \mathbb{R}^6$ and $\boldsymbol{\tau}_{\text{wave}} \in \mathbb{R}^6$ are forces and moments caused by wind and waves, respectively.

2.1.1 Reference Frames

A vessel maneuvers in 6 independent degrees of freedom (DOF), which are a combination of displacements and rotations. The combination completely describes the vessel pose [16]. The 6 DOF can be described using the body-fixed reference frame $\{b\} = (x_b, y_b, z_b)$. In this frame the x_b axis is fixed in the vessel forward direction, y_b axis is fixed in the vessel starboard side, and z_b is fixed in the vessel vertical axis, directed from top to bottom. The translational speed in the directions of the axes x_b, y_b, z_b are surge, sway and heave speed, denoted u, v, w respectively. The rotation rates about x_b, y_b, z_b are roll, yaw and pitch rate, denoted p, q, r, respectively. The vessel velocity is presented in Figure 2.1.

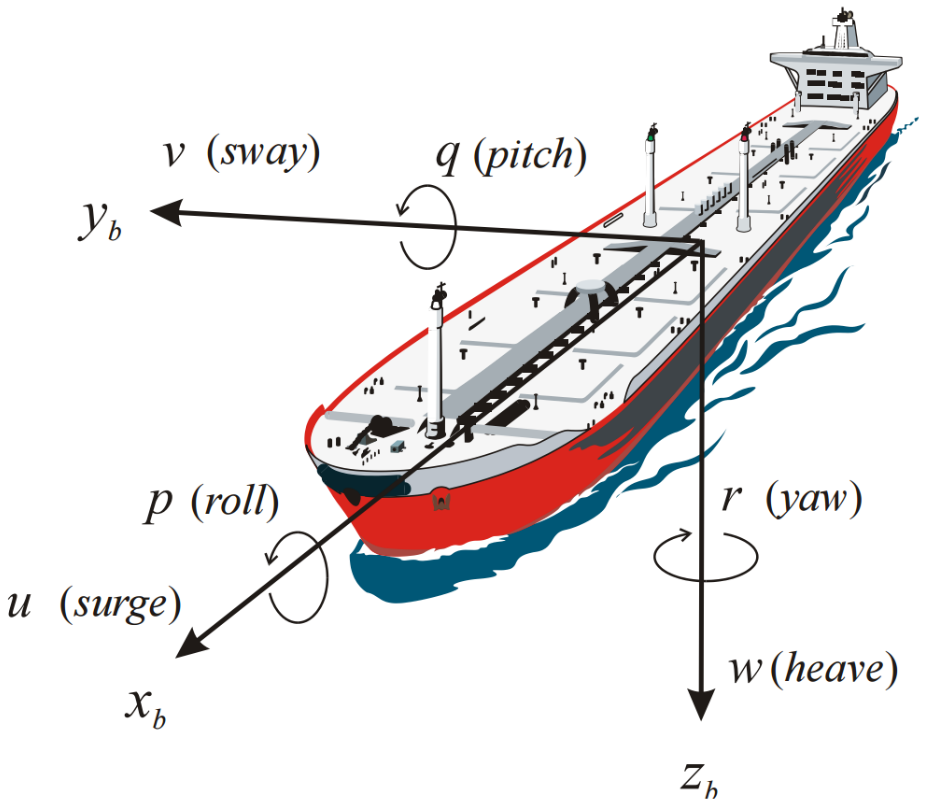


Figure 2.1: Motion in 6 DOF in the body-fixed reference frame [16].

The body-fixed reference frame gives no information about the vessel pose relative to the surrounding environment. To relate the body-fixed frame to the surrounding environments, the North-East-Down coordinate system $\{n\} = (x, y, z)$ is used as a reference frame, where x points towards true north, y points towards true east, and z points downwards normal to the surface of the Earth.

For navigation, we use a flat plane fixed on the surface of the Earth. One can assume that $\{n\}$ is inertial such that Newton's laws apply [16]. Since only local areas are considered, the NED-frame may be referred to as the world frame.

2.1.2 Euler Angle Transformations

Using two reference frames creates the need for a transformation between the velocity and pose in one frame to the other. The linear velocity transformation $\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})$ from {b} to {n} is given by

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) = \boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{x,\phi}, \tag{2.5}$$

where $\boldsymbol{R}_{z,\psi}, \boldsymbol{R}_{y,\theta}, \boldsymbol{R}_{x,\phi}$ are rotational matrices about z_b, y_b, x_b respectively. The vessel body-fixed velocity $\boldsymbol{v}_{b/n}^b$ can now be expressed in $\{n\}$ as

$$\dot{\boldsymbol{p}}_{b/n}^n = \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})\boldsymbol{v}_{b/n}^b \tag{2.6}$$

where $\boldsymbol{p}_{b/n}^n = [x, y, z]^\top$ is the position in the NED coordinate system. In a similar manner the Euler rate $\dot{\boldsymbol{\Theta}}_{nb} = \{\dot{\phi}, \dot{\theta}, \dot{\psi}\}^\top$ can be expressed by the transformation matrix $\boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})$, and the body-fixed angular velocity vector $\boldsymbol{\omega}_{b/n}^b = \{p, q, r\}^\top$ as

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}_{b/n}^b. \tag{2.7}$$

Summarizing these results, the kinematic equation can now be expressed as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_\Theta(\boldsymbol{\eta})\boldsymbol{v}, \tag{2.8}$$

which is equal to

$$\begin{bmatrix} \dot{\boldsymbol{p}}_{b/n}^n \\ \dot{\boldsymbol{\Theta}}_{nb} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_{b/n}^b \\ \boldsymbol{\omega}_{b/n}^b \end{bmatrix}.$$

2.1.3 Simplified ASV Model

To completely describe the vessel pose one needs to describe it in 6 DOF. For surface vessels, a normal assumption is that pitch, roll and heave may be neglected [16]. This leads to a 3 DOF representation, using surge, sway and yaw, which finally yields:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{2.9}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \tag{2.10}$$

where $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$ and $\boldsymbol{\eta} = [x, y, \psi] \in \mathbb{R}^2 \times SO(2)$, using SNAME notation, as presented in Table 2.1 [16]. In addition, $\boldsymbol{R}(\psi)$ is the simplified velocity transformation $\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})$ given in (2.5). The rotation matrix may now be expressed as

Table 2.1: 3 DOF standard notation from SNAME [34] for marine vessels.

DOF	**Description**	**Forces and moments**	**Velocities**	**Position and Euler angles**
1	motions in surge	X	u	x
2	motions in sway	Y	v	y
3	rotation in yaw	N	r	ψ

$$\boldsymbol{R}(\psi) \triangleq \boldsymbol{R}_{z,\psi} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.11}$$

Using a propeller and a rudder as actuators, the control input $\boldsymbol{\tau}$ may be modeled as:

$$\boldsymbol{\tau} \triangleq \boldsymbol{B}\boldsymbol{f}, \tag{2.12}$$

where $\boldsymbol{f}$ are the forces and moments from the propeller and rudder, and $\boldsymbol{B}$ describes the effect $\boldsymbol{f}$ has in each degree of freedom. The vector $\boldsymbol{f}$ and matrix $\boldsymbol{B}$ are given as:

$$\boldsymbol{f} = \begin{bmatrix} X \\ N \end{bmatrix}, \tag{2.13}$$

and

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{l_r} \\ 0 & 1 \end{bmatrix}, \tag{2.14}$$

where X is the force in surge and N the moment in yaw, caused by the propeller and rudder, respectively. In addition, l_r is the length from the rudder to the center of origin (CO). We model the yaw moment N as:

$$N = -K_{\delta_\psi} u^2 l_r \delta_\psi, \tag{2.15}$$

where u is the surge speed, and $K_{\delta_\psi} > 0$ is a rudder coefficient.

The inertia matrix in 3 DOF can be expressed as

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A = \begin{bmatrix} m & 0 & -my_g \\ 0 & m & mx_g \\ -my_g & mx_g & I_z \end{bmatrix} + \begin{bmatrix} -X_{\dot{u}} & -X_{\dot{v}} & -X_{\dot{r}} \\ -Y_{\dot{u}} & -Y_{\dot{v}} & -Y_{\dot{r}} \\ -N_{\dot{u}} & -N_{\dot{v}} & -N_{\dot{r}} \end{bmatrix}, \tag{2.16}$$

where $\boldsymbol{M}_{RB} \geq 0$ is the rigid body mass of the vessel, $\boldsymbol{M}_A \geq 0$ is the added mass, m is the weight of the vessel, and I_z is the inertia in the yaw rotation. In addition, x_g and y_g describes the center of gravity in the point (x_g, y_g, z_g) relative to origin of the body frame. A normal approximation for the added mass of surface vessels is to assume that the surge mode is independent of the steering dynamics [16]. This results in a simplified added mass matrix:

$$\boldsymbol{M}_A = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & N_{\dot{v}} & N_{\dot{r}} \end{bmatrix}. \tag{2.17}$$

We assume xz-plane symmetry and choose the origin of the body frame to be defined along the centerline of the vehicle. Hence, $y_g = 0$ and the rigid-body mass matrix can be simplified to

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix}. \tag{2.18}$$

Finally, the system inertia matrix can be expressed as:

$$\boldsymbol{M} = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{23} & m_{33} \end{bmatrix}. \tag{2.19}$$

The Coriolis and centripetal matrix $\boldsymbol{C}(\nu)$ is then given as :

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m_{23}r - m_{22}v \\ 0 & 0 & m_{11}u \\ m_{23}r + m_{22}v & -m_{11}u & 0 \end{bmatrix}. \tag{2.20}$$

The damping matrix $\boldsymbol{D}(\boldsymbol{\nu})$ for the system can be given as a sum of a linear damping matrix $\boldsymbol{D}_L$ and a non-linear damping matrix $\boldsymbol{D}_{NL}(\boldsymbol{\nu})$:

$$\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{D}_L\boldsymbol{\nu} + \boldsymbol{D}_{NL}(\boldsymbol{\nu})\boldsymbol{\nu}. \tag{2.21}$$

The damping matrix may be defined in several ways, based on which simplifications are made. The parameters for the vessel in the simulation in Section 5 is based on Loe's representation in [28], hence the definition from [28] of $\boldsymbol{D}_L$ and $\boldsymbol{D}_{NL}(\boldsymbol{\nu})$ is chosen. The matrices are defined as

$$\boldsymbol{D}_L = \begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix}, \tag{2.22}$$

and

$$\boldsymbol{D}_{NL}(\boldsymbol{\nu}) = \begin{bmatrix} X_{|u|u}|u| + X_{uuu}u^2 & 0 & 0 \\ 0 & Y_{|v|v}|v| + Y_{vvv}v^2 & 0 \\ 0 & 0 & N_{|r|r}|r| + N_{rrr}r^2 \end{bmatrix}. \tag{2.23}$$

The ASV experiences second order nonholonomic constraints also called nonintegrable velocity constraints. If the constraints had been holonomic, this could be verified by showing that the velocity constraints in (2.10) are holonomic (integrable). Firstly, the constraints must be partially integrable, which is verified by checking if (2.10) can be integrated into the form

$$\boldsymbol{g}(\boldsymbol{\nu}, \dot{\boldsymbol{\nu}}, t) = \mathbf{0}. \tag{2.24}$$

If this holds, (2.10) further integrability to a constraint on the form

$$\boldsymbol{f}(\boldsymbol{\nu}, t) = \mathbf{0} \tag{2.25}$$

is investigated. If such a constraint exists, (2.10) will be considered a holonomic constraint [30]. This check can be done by using Frobenius Theorem [21], as further described in [4, 30].

Constraints that are only partially integrable are first order nonholonomic, while constraints that are not partially integrable are second order nonholonomic. Parts of the requirements of Theorem 1 in [39] for vessel constraints to be partially integrable is $(\boldsymbol{C}_u(\boldsymbol{\nu}) + \boldsymbol{D}_u(\boldsymbol{\nu}))$ being constant. Where $\boldsymbol{C}_u(\boldsymbol{\nu})$ and $\boldsymbol{D}_u(\boldsymbol{\nu})$ are the Coriolis and centripetal, and damping matrices of the underactuated part of the vehicle, respectively. Clearly, this does not hold for (2.20), (2.22) and (2.23), which proves that (2.10) have second order nonholonomic constraints.

2.2 Motion Control and Guidance Systems

A guidance system is continuously calculating the desired position, velocity or acceleration of a vehicle. The desired states are used as reference values and will form either a setpoint, a trajectory, or a path. Based on the desired states a control objective is determined as a *setpoint regulation*, *trajectory-tracking control*, or *path-following control* [8]. Based on the control objective, a motion control system can be designed to satisfy the requirements. In *setpoint-regulation* the desired position and attitude are constant. In *trajectory-tracking* the systems tries to force the output $\boldsymbol{y}(t)$ to be equal to the time-variant desired output $\boldsymbol{y}_d(t)$. *Path-following* is following a predefined, time-invariant path.

2.2.1 Path Generation Using Waypoint Representation

A path for surface vehicles represented with waypoints is a set of Cartesian coordinates containing a start point, an end point and an ordered set of points in between. The set is expressed as,

$$WP = \{x_0 . y_0, x_1, y_1, \ldots, x_n, y_n\}, \tag{2.26}$$

where $n+1$ is the number of waypoints. Waypoints can have a speed and heading property, which are desired surge speed and heading of the vehicle when passing the given point. The waypoints are usually generated based on criterias, e.g. the path being feasible.

2.2.2 Line-of-Sight Guidance

A Line-of-Sight (LOS) guidance strategy usually consist of a reference point, an interceptor and a target. The reference point is usually stationary, while the target can either be stationary or moving. For this purpose the reference point and the target will both be waypoints, denoted as (x_k, y_k) and (x_{k+1}, y_{k+1}), respectively. This scenario is called a target-tracking, and its control objective is expressed as

$$\lim_{t\to\infty}[\boldsymbol{p}^n(t) - \boldsymbol{p}^n_t(t)] = \mathbf{0}, \tag{2.27}$$

where $\boldsymbol{p}^n_t = [x_t, y_t]^\top$ is the position of the target given in the world frame [8]. To define the steering laws in LOS we express the vehicle speed, denoted $U(t)$, and course angle, denoted $\chi(t)$, as

$$U(t) \triangleq= \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \geq 0 \tag{2.28}$$

$$\chi(t) \triangleq atan2(\dot{y}(t), \dot{x}(t)) \in \boldsymbol{S} \triangleq (-\pi, \pi]. \tag{2.29}$$

The operator $atan2(y, x)$ is the four-quadrant inverse tangent of y and x. By considering a straight line between the waypoints $\boldsymbol{p}_k = (x_k, y_k)$ and $\boldsymbol{p}_{k+1} = (x_{k+1}, y_{k+1})$, the angle of the straight line with respect to the NED-frame can be defined as

$$\alpha_k \triangleq \text{atan2}(y_{k+1} - y_k, x_{k+1} - x_k) \in \boldsymbol{S}. \tag{2.30}$$

The object for the LOS guidance strategy is to calculate the desired course angle χ_d, as illustrated in Figure 2.2. Using lookahead based steering the course angle is expressed as

$$\chi_d(e) = \chi_p + \chi_r(e), \tag{2.31}$$

where e is the cross-track error defined by a vector line from the vehicle, normal to the path line between the two waypoints. The parts of χ_d, χ_p and χ_r, are expressed as

$$\chi_p = \alpha_k \tag{2.32}$$

and

$$\chi_r(e) \triangleq arctan\Big(\frac{-e}{\Delta}\Big), \tag{2.33}$$

respectively. This ensures that the vehicle course is directed towards the path, using the tuning parameter $\Delta > 0$ as look-ahead distance [8].

For path-following controllers, the desired course angle χ_d is transformed to the desired heading angle expressed as

$$\psi_d = \chi_d - \beta, \tag{2.34}$$

where β is the sideslip angle of the vehicle, expressed as [16]:

$$\beta = arcsin\Big(\frac{v}{U}\Big). \tag{2.35}$$

Finally, the desired course angle ψ_d from the LOS-algorithm is used as a reference in a heading controller.

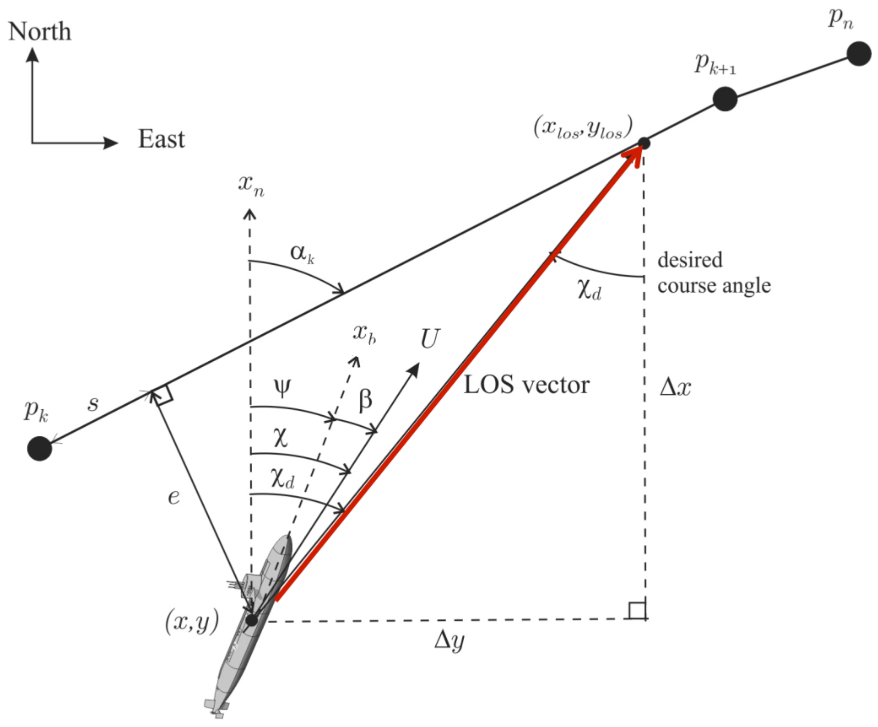


Figure 2.2: 2-D LOS principle [16].

2.2.3 Path Tracking for Marine Surface Vessels

The LOS guidance strategy is dependent on the planned path only containing straight lines between waypoints. However, in numerous scenarios, it is necessary that planned paths are curved. By using curved and parametrized paths, it is possible to plan a feasible path for marine surface vessels. A parametrized path is defined as a geometric curve parametrized by a continuous path variable [33].

The 2D path following method for marine surface vessels presented in [7] will be used for tracking globally planned trajectories in Chapter 4. The method is designed for a fully actuated vessel but is compatible with underactuated vessels if sideslip compensation and a dynamic controller state are introduced.

A geometrical path defined on a two-dimensional Euclidean plane, parametrized by a scalar value $\theta \in \mathbb{R}$, is introduced as the desired path. For any value of θ, the inertial position of the path is expressed as:

$$\boldsymbol{p}_d(\theta) = \begin{bmatrix} x_d \\ y_d \end{bmatrix}, \tag{2.36}$$

Where x_d and y_d are desired displacement along the north and east axis, respectively. A point mass particle with inertial position $\boldsymbol{p} \in \mathbb{R}^2$ and velocity $\mathbf{v} \in \mathbb{R}^2$ is desired to reach the path and stay on it. The size and orientation of $\mathbf{v}$ are denoted as

$$U = \|\mathbf{v}\|_2 = (\mathbf{v}^\top \mathbf{v})^{\frac{1}{2}} \tag{2.37}$$

and

$$\chi = \arctan\left(\frac{v_y}{v_x}\right), \tag{2.38}$$

respectively. By controlling U and χ, the particle $\boldsymbol{p}$ can be forced to converge to and stay on the desired path.

For a given θ, a new reference frame denoted Path Parallel (PP) frame is defined. The angle of the reference frame relative to the inertial frame is given as:

$$\chi_t(\theta) = \arctan\left(\frac{y'_d(\theta)}{x'_d(\theta)}\right), \tag{2.39}$$

where $x'_d(\theta) = \frac{dx_d}{d\theta}$ and $y'_d(\theta) = \frac{dy_d}{d\theta}$. The error vector between the particle point $\boldsymbol{p}$ and point $\boldsymbol{p}_d(\theta)$ expressed in the PP frame is given as:

$$\boldsymbol{\epsilon} = \boldsymbol{R}_p^\top(\chi_t)(\boldsymbol{p} - \boldsymbol{p}_d(\theta)), \tag{2.40}$$

where $\boldsymbol{R}_p(\chi_t) \in SO(2)$ is the rotation matrix from the inertial frame to the PP frame. As seen in Figure 2.3, the error consists of two parts. The cross track error e and the along-track error s denoted the lateral and longitudinal distance, respectfully. Hence, the error vector is defined as

$$\boldsymbol{\epsilon} = [s, e]^\top. \tag{2.41}$$

A path tangential speed is introduced as U_{PP} and is given as the velocity of the PP frame with respect to the inertial frame, given in the PP frame. The path tangential speed is considered as a virtual input for stabilizing s, and is set by choosing U_{PP} as:

$$U_{PP} = U\cos(\chi - \chi_t) + \gamma s, \tag{2.42}$$

where $\gamma > 0$ is a gain parameter. Since θ is the controllable path parameter a relationship between θ and U_{PP} is defined and expressed as:

$$\dot{\theta} = \frac{U_{PP}}{\sqrt{x_d'^2 + y_d'^2}} = \frac{U\cos(\chi - \chi_t) + \gamma s}{\sqrt{x_d'^2 + y_d'^2}}. \tag{2.43}$$

To stabilize the cross track error e, the course χ of the particle is controlled. The particle is assumed to be able to change value of U and χ instantaneously, hence

$$\chi = \chi_d. \tag{2.44}$$

The angular difference between the desired course path and the path tangential course is expressed as

$$\chi_r = \chi_d - \chi_t, \tag{2.45}$$

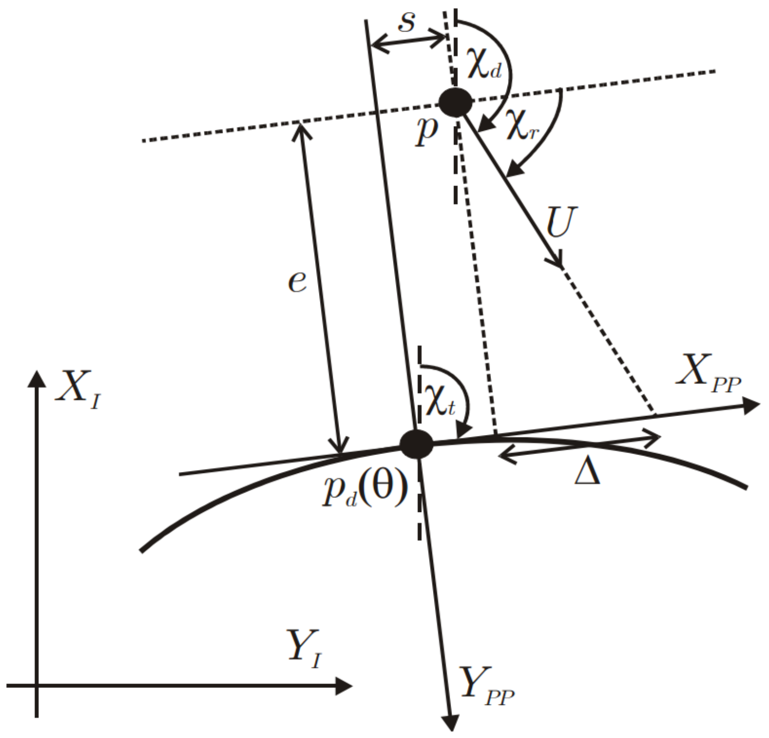


Figure 2.3: 2-D path following for marine vessels [7].

where we define

$$\chi_r(e) = \arctan\left(-\frac{e}{\Delta}\right), \tag{2.46}$$

where $\Delta > 0$ is the lookahead distance. The desired course angle χ_d is expressed as:

$$\chi_d(\theta, e) = \chi_t(\theta) + \chi_r(e). \tag{2.47}$$

Finally, it is proved in [7] that the origin $\boldsymbol{\epsilon} = \mathbf{0}$ is uniformly globally asymptotically and locally exponential stable (UGAS/ULES) if θ is updated by (2.43), χ is equal to (2.47) and U is non-zero and have a lower bound.
By using the sideslip compensation when calculating the desired heading angle, we get:

$$\psi_d = \chi_d - \beta, \tag{2.48}$$

where β is the sideslip of the vessel.

The methods introduced in Chapter 4 use a modified version of the introduced path following method to perform trajectory tracking.

2.3 Spaces In Motion Planning

In motion planning and collision avoidance, different spaces are used to describe the state parameters of a vehicle and its surrounding environment. For this purpose, the work space and the configuration space will be utilized.

The work space is a description of the environment the vehicle is in, and the obstacles in the environment. A two-dimensional environment may be expressed by the work space $\mathcal{W} \in \mathbb{R}^2$ [3]. Obstacles in $\mathcal{W}$ may be defined as the set $\mathcal{O} = \{O_1, O_2, \dots, O_n\}$, where O_i is an obstacle in the workspace. For collision avoidance a space named the free work space is defined $\mathcal{W}_{free} = \{\boldsymbol{\eta} \in \mathcal{W} \mid \boldsymbol{\eta} \notin \mathcal{O}\}$. By denoting the footprint of a vehicle in $\mathcal{W}$ as $\mathcal{A}(\boldsymbol{\eta})$, any configuration $\boldsymbol{\eta} = [x, y, \psi]^\top$ where $\mathcal{A}(\boldsymbol{\eta}) \in \mathcal{W}_{free}$ will be a safe position. The vehicle is intersecting an obstacle if $\mathcal{A}(\boldsymbol{\eta}) \cap \mathcal{O} \neq \emptyset$ is true, and any configuration where this holds will therefore be forbidden.

In general, the configuration of a vehicle with a certain number DOFs is given by at least a corresponding number of parameters [36]. To describe this in a space, it is not sufficient to use the workspace notation. E.g. for the 3 DOF vessel used in (2.9), one would at least need a three-dimensional space to fully describe the configuration. The configuration space expressed as $\mathcal{C} \in \mathbb{R}^2 \times SO(2)$ [36] completely describes the ASV pose.

Compared to the work space where a vehicle is described by its extent, a configuration in the configuration space is given as a single point $\boldsymbol{\eta}$, specified by the vehicle state parameters [3]. As in the work space, there are forbidden configurations. In the configuration space, these are defined as the set of all configurations where the vehicle intersects with the obstacle set $\mathcal{O}$, and is expressed as

$$\mathcal{C}_{forb} = \big\{\boldsymbol{\eta} \in \mathcal{C} \mid \mathcal{A}(\boldsymbol{\eta}) \cap \mathcal{O} \neq \emptyset\big\}. \tag{2.49}$$

Any other point in $\mathcal{C}$ will be free, and will define the set

$$\mathcal{C}_{free} = \{\boldsymbol{\eta} \in \mathcal{C} \mid \mathcal{A}(\boldsymbol{\eta}) \cap \mathcal{O} = \emptyset\}. \tag{2.50}$$

The extent of a vehicle might be complex and $\mathcal{C}_{free}$ are therefore hard to calculate. By approximating the vehicle footprint as a circle with radius equal to the largest distance from the center of the vehicle to any other point, we can simplify the configuration space (inspired by [3, 13]).

If a vehicle is represented as a single point, the configuration space is independent of the vehicle heading. Hence, a simplified configuration space can be defined as $\bar{\mathcal{C}} \in \mathbb{R}^2$. A surface vehicle with position (x, y) and heading ψ may therefore be represented in the simplified configuration space in two dimensions, only using position parameters (x, y). Configurations in the new configuration space may now be defined as $\boldsymbol{p} = [x, y]^\top$. Any point $\boldsymbol{p}$ closer to an obstacle O_i than the radius length l are considered to be part of $\bar{\mathcal{C}}_{forb}$, as shown in Figure

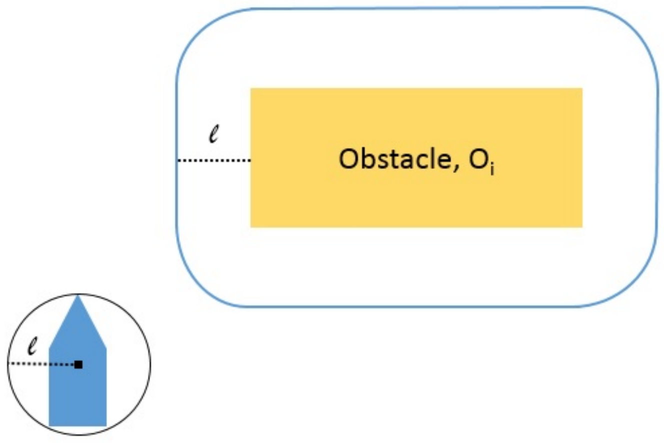


Figure 2.4: Circular approximation of the vessel footprint, extending the forbidden area around the obstacle O_i with the radius l of the circle.

2.4. The footprint of the vehicle with configuration $\boldsymbol{p}$ may now be given by $\bar{\mathcal{A}}(\boldsymbol{p})$, and will define the circle with radius l, centered in $\boldsymbol{p}$. The new free and forbidden space in $\mathbb{R}^2$ may now be expressed as

$$\bar{\mathcal{C}}_{free} = \{\boldsymbol{p} \in \bar{\mathcal{C}} \mid \bar{\mathcal{A}}(\boldsymbol{p}) \cap \mathcal{O} = \emptyset\} \tag{2.51}$$

and

$$\bar{\mathcal{C}}_{forb} = \{\boldsymbol{p} \in \bar{\mathcal{C}} \mid \bar{\mathcal{A}}(\boldsymbol{p}) \cap \mathcal{O} \neq \emptyset\}, \tag{2.52}$$

respectively. The simplified configuration space $\bar{\mathcal{C}}$ with points $\boldsymbol{p}$ and footprint $\bar{\mathcal{A}}(\boldsymbol{p})$ is used for the simulations in Chapter 5.

2.4 Reactive COLAV Methods

Reactive COLAV methods, also called local or sense-act methods, base their actions on currently available sensor data. These methods are memoryless and consider only the immediate environment, which leads to a low computational cost. This makes the local methods excellent for real-time systems and dynamic environments [28].

A major drawback for reactive methods is the lack of ability to produce an optimal path to the goal [17]. They are less likely to find a path to the goal if there exist one, compared to global methods. This is due to the reactive methods significant risk of getting stuck in a local minima, and the lack of stored environment data.

The most commonly used reactive COLAV approaches can be divided into directional and velocity space based methods [31]. The directional method searches for direction for the vehicle to head in without taking the vehicle dynamics into account. Vector Field Histogram [5] and Potential field method [23] are examples of common directional COLAV methods. The velocity COLAV approaches searches in the velocity space, which makes the methods able to take the dynamic and kinematic constraints into account. The Dynamic Window (DW) algorithm [17] and Velocity Obstacles [15] are examples of velocity space based reactive COLAV methods.

2.4.1 Dynamic Window (DW) Algorithm

The DW algorithm, presented by Fox [17] in 1997, is a reactive COLAV algorithm searching for an optimal solution within the velocity space. The velocity space is formed by the vehicle translational speed and rotational rate. For an ASV, a point in the velocity space consists of the surge speed u and the yaw rate r, which form a velocity pair (u, r), while the sway speed v is not considered. The algorithm generates a search space considering multiple sets of constraints in the velocity space. The search space is discretized into a finite set of velocity pairs. For every velocity pair within the search space, the DW algorithm predicts what trajectory the pair will lead to for the vehicle. Finally, an objective function finds the optimal velocity pair based on the predicted trajectory.

The original DW algorithm is intended for vehicles with first-order nonholonomic constraints. The search space of the algorithm is generated by the union of the following sets:

The DW algorithm considers only *admissible velocities*. For a velocity pair to be considered admissible, the vehicle has to be able to stop before colliding with the closest obstacle along the predicted trajectory defined by the pair. By defining $\dot{u}_{min}$ and $\dot{r}_{max}$ as the accelerations for breakage, the set V_a of admissible velocities for a ship is defined as:

$$
\begin{aligned}
V_a = \Big\{(u, r) \mid u &\leq \sqrt{2 \cdot dist(u, r) \cdot |\dot{u}_{min}|} \\
\wedge |r| &\leq \sqrt{2 \cdot dist(u, r) \cdot |\dot{r}_{max}|}\Big\},
\end{aligned}
\tag{2.53}
$$

where $\mathrm{dist}(u, r)$ is the distance to the closest obstacle on the curvature [17].

The physical constraints on the vehicle motors cause the acceleration within a short time step to be limited. These constraints form the dynamic window, which is a set of all velocity pairs reachable from the vehicle current velocity within a given short time interval. Let (u^*, r^*) be the current velocity and let

$\dot{u}_{max}$ and $\dot{r}_{max}$ be the maximum possible accelerations applied during a time step T_s. Then the dynamic window V_d is defined as:

$$V_d = \big\{(u,r) \mid u \in [u^* + \dot{u}_{min} \cdot T_s, u^* + \dot{u}_{max} \cdot T_s] \wedge r \in [r^* \pm \dot{r}_{max} \cdot T_s]\big\}. \quad (2.54)$$

The maximum possible translational and rotational velocities, u_{max} and r_{max}, respectively, creates a static window V_s in the velocity space. This set of *possible velocities* is defined as:

$$V_s = \big\{(u,r) \mid u \in [0, u_{max}] \wedge r \in [-r_{max}, r_{max}]\big\}. \quad (2.55)$$

Lastly, the search space V_r is defined by the union of the three sets

$$V_r = V_s \cap V_a \cap V_d, \quad (2.56)$$

and discretized into a finite set of velocity pairs.

The trajectory prediction for each velocity pair is made by assuming that the vehicle accelerates instantly to the velocity given by the pair (u,r). The velocity pair creates a curvature with the radius C_r given as [17]

$$C_r = \frac{u}{r}, \quad (2.57)$$

as shown in Figure 2.5.

An optimized velocity pair will then be chosen by the following objective function:

$$\begin{array}{l} \max\limits_{(u,r)} G(u,r) = \sigma(\alpha \cdot heading(u,r) + \beta \cdot dist(u,r) + \gamma \cdot vel(u,r)) \\ s.t.\ (u,r) \in V_r \end{array}, \quad (2.58)$$

where the three functions, heading, dist and vel are based on the velocity pair and the corresponding predicted trajectory. The functions will be normalized to a value in the range [0,1], and weighted by the variables $\alpha, \beta, \gamma > 0$, and σ is a low pass filter for smoothing the function.

Heading(u,r) is a value based on the vehicle direction towards the goal, given by $180\ ^\circ - \theta$. The angle θ is given by the angle of the goal relative to the vehicle heading direction. This direction changes with different velocity pairs, so θ must be calculated from a predicted pose, which is determined by assuming that the vehicle follows the planned trajectory while applying maximal deceleration. The predicted stopping pose will then be used to find θ. The heading function leads to a more smooth maneuvering towards goal once the obstacle is passed

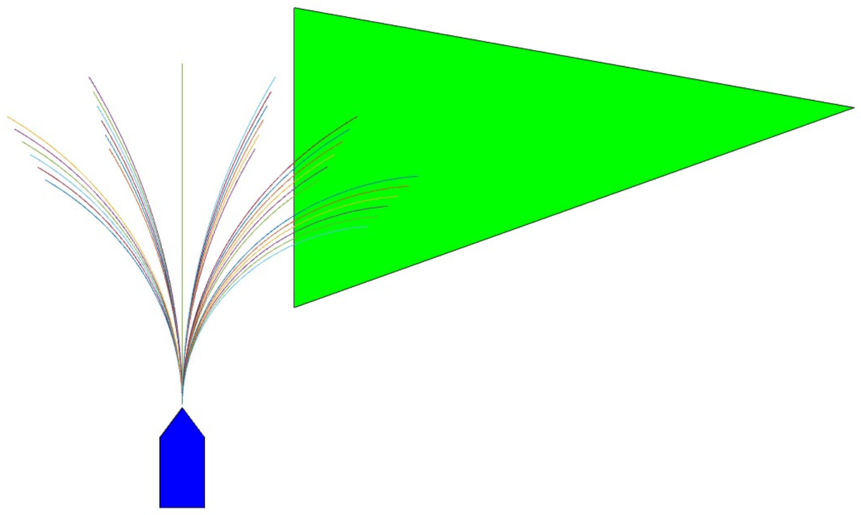


Figure 2.5: Predicted trajectories for the velocity pairs in the discrete search space. The solid colored lines are predicted trajectories.

[17].

Dist(u, r) is a value based on the distance of the closest obstacle. If there are any obstacles on the predicted path, then *Dist* is valued as the distance to the closest obstacle. If not, *dist* will be set to a relatively large constant.

Vel(u, r) is the velocity of the vehicle along the predicted trajectory and is solely determined by u. This value and *heading* will ensure that the vehicle values progress towards the goal, even when circumventing an obstacle.

2.5 Deliberate COLAV Methods

Deliberate COLAV methods, also called global or sense-plan-act, are methods that find a route from a given start to a given goal based on available environment data. The goal can either be a vehicle position or a more detailed description of the end state [28]. Global methods are usually not as prone to getting trapped in a local minima as reactive methods, and can often guarantee that a path to the goal will be found if one exists (completeness). However, numerous methods offer only probabilistic completeness, but will in return offer a superior asymptotic convergence rate [6] (e.g. RRT).

Global methods suffer a major challenge in real-time systems due to long computational time compared to reactive methods. This may cause problems when unexpected situations occur, and decisions might have to be taken rapidly.

The A* algorithm [19] is a deliberate method that offers optimality and completeness but does not take the vehicle dynamics into account. The RRT algorithm [27] is only probabilistic complete and likely to generate a less optimal path than the A* algorithm. Unlike the A* algorithm, the RRT algorithm can take the vehicle dynamics into account, which enables it to plan a feasible trajectory for the vehicle. The A* algorithm is applied to an ASV in [26], where it handles collision avoidance for static obstacles. We wish to implement a hybrid COLAV method that handles both static and dynamic obstacles, hence the deliberate method is required to return a feasible trajectory. Consequently, we use the RRT algorithm as a global trajectory planner in the hybrid COLAV method.

2.5.1 Rapidly-Exploring Random Trees (RRT)

The Rapidly-Exploring Random Tree path planning algorithm introduced by LaValle [27] in 1998 can be used as a deliberate COLAV algorithm searching for a global path leading to the goal. The algorithm is neither complete or optimal but is, however, probabilistic complete under general conditions and scales well with high numbers of DOFs. The RRT approach takes the vehicle dynamics into account and can be directly applied to nonholonomic and kinodynamic planning to generate paths which are feasible for the vehicle. This makes it well suited for use on AUVs modeled in 3 DOFs [37], and is in [28] shown to be well suited for ASVs as well.

The RRT algorithm searches randomly through the state space to obtain paths leading towards the goal. If the search is finished and a path leading to goal have been found, the optimal path in the tree is chosen based on chosen metrics. Although the search is randomized, it is heavily biased towards unexplored areas of the state space and is therefore likely to find a path to goal if there exists one. An example of a complete RRT that considers the ASV dynamics is displayed in Figure 2.6.

The Rapid-Expanding Random search tree $\boldsymbol{\Upsilon}$ is generated such that all of the vertices are states in $\bar{\boldsymbol{\mathcal{C}}}_{free}$ and the edges are paths lying in $\bar{\boldsymbol{\mathcal{C}}}_{free}$. The initial vertex is the current state of the vehicle $\mathbf{x}_{init}$. To expand the tree, a random state $\mathbf{x}_{rand} \in \bar{\boldsymbol{\mathcal{C}}}_{free}$ is chosen by *RANDOM_STATE*() and a function *NEAREST_NEIGHBOUR*($\mathbf{x}_{rand}, \boldsymbol{\Upsilon}$) is used to find the vertex $\mathbf{x}_{near} \in \boldsymbol{\Upsilon}$ closest to $\mathbf{x}_{rand}$. A function *SELECT_INPUT* ($\mathbf{x}_{rand}, \mathbf{x}_{near}$) is then used to find the input $\boldsymbol{u}$ that yields the shortest feasible path from $\mathbf{x}_{near}$ to $\mathbf{x}_{rand}$ and ensures that the vehicle state stays in $\bar{\boldsymbol{\mathcal{C}}}_{free}$. If the path resides in $\bar{\boldsymbol{\mathcal{C}}}_{free}$ for a time period Δt, the path is collision free and added to Υ. This is done in *NEW_STATE*($\mathbf{x}_{near}, \boldsymbol{u}, \Delta t$) by applying $\boldsymbol{u}$ to the vehicle for the time Δt yielding a new vertex $\mathbf{x}_{new}$ and an

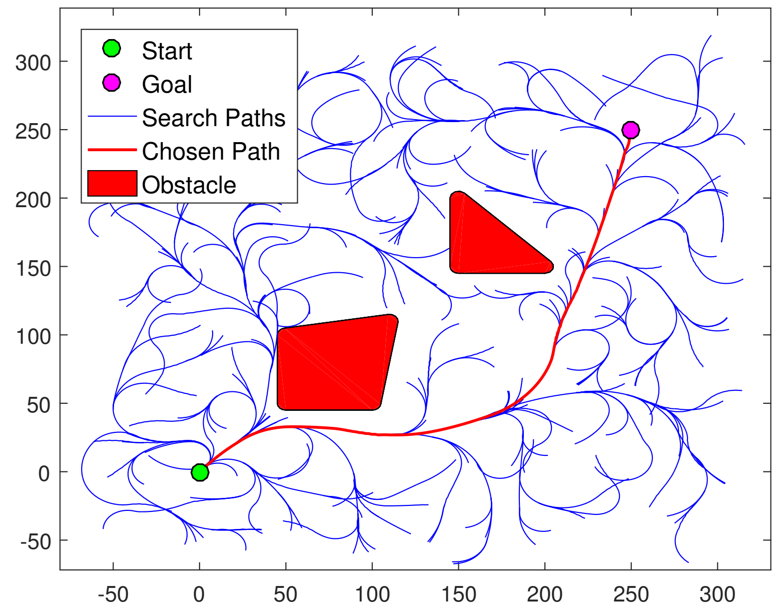


Figure 2.6: Randomly-Exploring Random Tree.

edge from $\mathbf{x}_{near}$ to $\mathbf{x}_{new}$. If the path does not reside inside $\bar{\boldsymbol{\mathcal{C}}}_{free}$ no new vertex is added to $\boldsymbol{\Upsilon}$ and the next iteration of the loop is started. The loop continues until a maximum number$K$ of vertices are added to $\boldsymbol{\Upsilon}$. A comprehensible overview of the general RRT algorithm is given in Algorithm 1 [27].

Algorithm 1: Generate RRT

```
1:  $\boldsymbol{\Upsilon} \leftarrow INIT(\mathbf{x}_{init})$
2:  for $k = 1$ to $K$ do
3:      $\mathbf{x}_{rand} \leftarrow RANDOM\_STATE()$
4:      $\mathbf{x}_{near} \leftarrow NEAREST\_NEIGHBOUR(\mathbf{x}_{rand}, \boldsymbol{\Upsilon})$
5:      $\mathbf{x} \leftarrow SELECT\_INPUT(\mathbf{x}_{rand}, \mathbf{x}_{near})$
6:      $\mathbf{x}_{new} \leftarrow NEW\_STATE(\mathbf{x}_{near}, \boldsymbol{u}, \Delta t)$
7:      if $COLLISION\_FREE(\mathbf{x}_{near}, \mathbf{x}_{new}, \boldsymbol{u}, \Delta t)$ then
8:          $\boldsymbol{\Upsilon} \leftarrow ADD\_VERTEX(\mathbf{x}_{new})$
9:          $\boldsymbol{\Upsilon} \leftarrow ADD\_EDGE(\mathbf{x}_{near}, \mathbf{x}_{new}, \boldsymbol{u}, \Delta t)$
10:     end if
11: end for
12: return $\boldsymbol{\Upsilon}$
```

The functions $COLLISION\_FREE(\mathbf{x}_{near}, \mathbf{x}_{new}, \boldsymbol{u}, \Delta t)$ and $SELECT\_INPUT(\mathbf{x}_{rand}, \mathbf{x}_{near})$ make sure the new state of the algorithm is reachable and feasible, and are hence the most crucial functions in the algorithm. Due to these functions, any vertices added to $\boldsymbol{\Upsilon}$ are guaranteed to be reachable from the initial state if the vehicle model is accurate. Note that the feasibility of the algorithm is dependent on the accuracy of the model, hence the algorithm may not be able to guarantee completely feasible paths.

Performance Enhancement

By adding a bias towards the goal, the RRT performance can be improved significantly [37]. This can be done by letting the $RANDOM\_STATE$ return the goal state with a probability based on a bias coefficient. If the function does not return the goal state, a random state is returned. This guarantees a progress towards goal when the RRT searches the state space.

To obtain convergence when using randomized algorithms, a hybrid system of a complete algorithm and a randomized algorithm can be implemented [18]. By using this method, the algorithm can consider the dynamic constraints of the vehicle. In [28], the A* algorithm [19] has been used as a guide for the RRT algorithm, which yields a great performance gain.

2.6 Hybrid COLAV methods

In real-time decision making, the rapid computations of the reactive methods are needed. However, the lack of knowledge beyond the vehicle's immediate environment causes reactive methods to be likely to make poor path choices when trying to reach the goal. Deliberate methods will exploit knowledge of the environment and are more likely to make good path choices that do not trap the vehicle in a local minima.

Hybrid methods exploit the benefits from both reactive and deliberate methods. The methods are built as a hierarchy with deliberate algorithms in the global layer at the top to ensure that the vehicle is likely to follow a path leading towards the goal. A local layer is implemented for rapid real-time responses on events not planned by the global method. On the lowest level, the vehicle controllers ensure that the vehicle is following the trajectory planned by the reactive layer. The goal of the hybrid method is to have a deliberate method plan trajectories with given desired properties, which will be used as a guide for the reactive method. The trade-off between probability to find the goal (completeness) and response time (responsiveness) is illustrated in Figure 2.7.

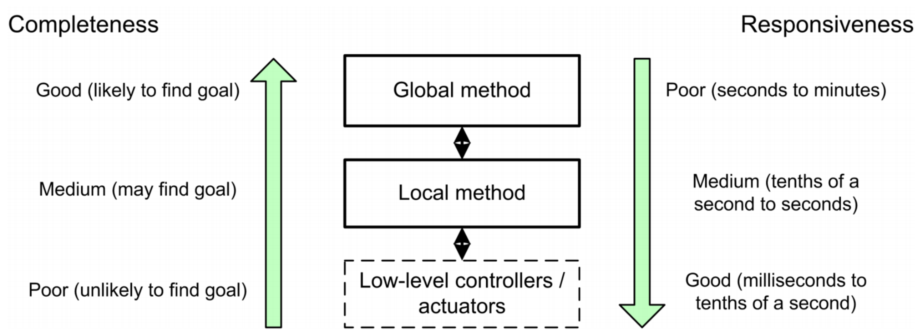

Completeness
Responsiveness
Good (likely to find goal)
Medium (may find goal)
Poor (unlikely to find goal)
Global method
Local method
Low-level controllers / actuators
Poor (seconds to minutes)
Medium (tenths of a second to seconds)
Good (milliseconds to tenths of a second)

Figure 2.7: Hybrid COLAV methods exploit the local method's low computation time and the completeness of the global method. Courtesy of [28].

2.7 COLREGS

The International Regulations for Preventing Collisions at Sea 1972 (COLREGS) are published by the International Maritime Organization. These regulations are set out to be followed by vessels at sea to prevent collisions between each other. COLREGS consist of 38 rules divided into different parts which are further divided into subsections [10].

For reactive COLAV methods for ASVs, the most relevant rules are in part B - Steering and sailing, under Section II (Conduct of vessels in sight of one another). The most relevant rules when considering ASVs may be summarized as following [10]:

Rule 13. Overtaking - the overtaking vessel should keep out of the way of the vessel being overtaken.

Rule 14. Head-on situations - Two power-driven vessels meeting on reciprocal or nearly reciprocal courses each shall alter her course to starboard so that each shall pass on the port side of the other.

Rule 15. When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel.

COLREGS 13, 14 and 15 are previously applied to COLAV methods using Velocity Obstacle (VO) [25] and by using multiobjective optimization with interval programming (IVP) [2].

Chapter 3

Modifications to the DW Algorithm

The original Dynamic Window algorithm in [17] is intended for vehicles with first-order nonholonomic constraints. As presented in Section 2.1.3, ASVs has second order nonholonomic constraints which do not fit with the assumptions done when designing the DW algorithm.

A modification to the DW algorithm for horizontal collision avoidance for AUVs is proposed in [12, 13], and denoted Algorithm A. In [13, 32], algorithm A described in Section 3.1 is compared with the original DW algorithm for use on AUVs and ASVs. Through simulations, it was concluded that algorithm A was the better option for use on AUVs and ASVs. However, suggestions for further development of the algorithm are proposed.

The distance function in Algorithm A is solely dependent on the time until reaching the avoidance region (3.13) along a trajectory. Hence, when the vehicle is inside the avoidance region, the distance function of every velocity pair is zero. If the distance function is ruled out, the cost function only considers the ASV speed and progress towards the goal, not whether a trajectory leads towards an obstacle or not. This motivates modifying Algorithm A so that the vehicle is desired to leave the avoidance region if entered. Two new algorithms, algorithms B and C, addressing this problem are presented in this section and compared through simulations presented in Section 6.1.2.

3.1 Algorithm A

As ASVs, AUVs have second order nonholonomic constraints and time-varying acceleration limits, due to the non-linear responses caused by the rudder. The modifications in Algorithm A are proposed for AUVs with a constant depth, neglecting the roll and pitch angles. The use of a modified search space and a new trajectory prediction method reduces the mean square error for AUVs to approximately one percent, compared to error of the original DW algorithm. It is argued through mild assumptions that the same representation holds for an ASV, as for the AUV in [12, 13]. In addition, Algorithm A is tested for ASVs in [32] and yields significantly better results than the original DW algorithm. One will therefore expect similar performance improvements for ASVs, as the improvements Eriksen [12] presented for AUVs. However, the AUV used when testing the algorithm is modelled with only linear damping, unlike the non-linear damping matrix (2.21) in the ASV model (2.2). Due to the non-linearity of the damping matrix, the simulation results might differ from the results in [12, 13]. A further modification to account for the non-linearity is presented in this section.

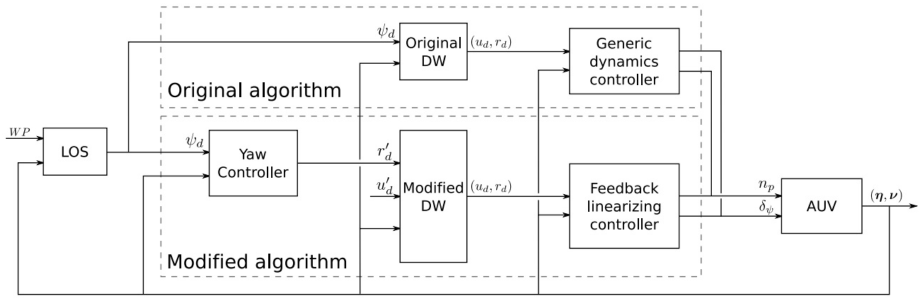


Figure 3.1: Architecture overview of original DW algorithm and Algorithm A. Courtesy of [12].

Figure 3.1 presents an overview of the architecture of the original DW algorithm and Algorithm A. One can see that Algorithm A takes a desired velocity pair (r'_d, u'_d) as input, where r'_d is decided from the output of the yaw controller. The original DW algorithm, however, depends on desired heading ψ_d directly. The desired yaw rate r'_d is found by parsing the output ψ_d of the LOS block into a yaw-controller by using the control law

$$r'_d = -k_\psi(\psi - \psi_d) + \dot{\psi}_d, \tag{3.1}$$

where $k_\psi > 0$ is a constant gain. The introduction of r'_d and u'_d enables external control of the surge speed and yaw rate.

3.1.1 Modified Search Space and Objective Function

The original search space does not take the actuator model and limitations into account. To ensure that the velocity pair chosen by the algorithm is feasible, a small time interval T_a is defined for changing the rudder angle δ_ψ. The time interval T_a is smaller than the DW period T_s used when calculating the search space, thus $T_a < T_s$. The rudder angle and angle rate is constrained by

$$\|\delta_{max}\|_\infty \leq \delta_{max} \tag{3.2}$$

and

$$\left\|\dot{\delta}_{max}\right\|_\infty \leq \dot{\delta}_{max}, \tag{3.3}$$

respectively, which give the possible values for δ_ψ during the time interval T_a:

$$\delta_\psi \in sat\Big(\Big[\delta_\psi^* - T_a\dot{\delta}_{max}, \delta_\psi^* + T_a\dot{\delta}_{max}\Big], \delta_{max}\Big), \tag{3.4}$$

where $sat(\cdot)$ is the saturation function, and ψ^* is the current rudder angle. The actuation in surge speed has constraints given by

$$F_X \in [F_{X,min}, F_{X,max}]. \tag{3.5}$$

The forces and moments working on the vehicle can then be described by $\boldsymbol{\tau}(\boldsymbol{\nu}, \delta_\psi, F_X)$, given in Section 2.1. The limits on the acceleration $\dot{\boldsymbol{\nu}}$ can be found from

$$\dot{\boldsymbol{\nu}}_i = \boldsymbol{M}^{-1}(\boldsymbol{\tau}_i - \boldsymbol{C}(\boldsymbol{\nu}^*)\boldsymbol{\nu}^* - \boldsymbol{D}(\boldsymbol{\nu}^*)\boldsymbol{\nu}^*), \tag{3.6}$$

where $i \in \{min, max\}$,

$$\boldsymbol{\tau}_{min} \triangleq \boldsymbol{\tau}(\boldsymbol{\nu}^*, max(\delta_\psi), min(F_X)), \tag{3.7}$$

and

$$\boldsymbol{\tau}_{max} \triangleq \boldsymbol{\tau}(\boldsymbol{\nu}^*, min(\delta_\psi), max(F_X)). \tag{3.8}$$

By using these limits, the modified Dynamic Window V_d can be expressed as:

$$\begin{aligned} V_d = \big\{(u,r) \mid u \in [u^* + \dot{u}_{min} \cdot T_s, u^* + \dot{u}_{max} \cdot T_s] \\ \wedge\, r \in [r^* + \dot{r}_{min} \cdot T_s, r^* + \dot{r}_{max} \cdot T_s]\big\}, \end{aligned} \tag{3.9}$$

which unlike (2.54) is not required to fulfill $\dot{r}_{max} = -\dot{r}_{min}$.

The set of possible velocities V_s is found with respect to the actuator saturation limits. A function g(u,r) is defined, which is positive semi-definite for feasible velocities, and negative otherwise. The set of possible solutions is expressed using the function g(u,r) as:

$$V_s = \Big\{(u,r) \mid g(u,r) \geq 0\Big\}. \tag{3.10}$$

The function g(u,r) is numerically computed by finding the boundaries of the steady state solution of the dynamics, given as:

$$\boldsymbol{M}\dot{\boldsymbol{\nu}}_r = \boldsymbol{\tau}(\boldsymbol{\nu}_r, \delta_\psi, F_X) - \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = 0, \tag{3.11}$$

for feasible pairs of rudder position and surge actuation. The set of possible velocities will now make a cone shape, while the Dynamic Window will still be formed as a rectangle. A new set of dynamically feasible velocities V_f is defined as

$$V_f = V_d \cap V_s. \tag{3.12}$$

To define the new set of admissible velocities V_a, two new regions which further penalizes trajectories leading towards an obstacle are introduced. This is done by using the circle approximation of the vehicle footprint, described in Section 2.3. The two regions are defined as the avoidance region

$$\boldsymbol{\Omega} \triangleq \left\{ \boldsymbol{p} \in \bar{\boldsymbol{\mathcal{C}}} \middle| \left\| \boldsymbol{p} - \boldsymbol{p_{forb}} \right\|_2 \leq r_{\boldsymbol{\Omega}} \right\}, \tag{3.13}$$

and the antitarget region

$$\boldsymbol{\mathcal{T}} \triangleq \left\{ \boldsymbol{p} \in \bar{\boldsymbol{\mathcal{C}}} \middle| \left\| \boldsymbol{p} - \boldsymbol{p_{forb}} \right\|_2 \leq r_{\boldsymbol{\mathcal{T}}} \right\}, \tag{3.14}$$

where $\boldsymbol{p}_{forb} \in \mathbb{R}^2$ is the closest point in the forbidden configuration space $\bar{\boldsymbol{\mathcal{C}}}_{forb}$ expressed in (2.52), $r_{\boldsymbol{\mathcal{T}}} \geq 0$ is the radius of the approximated footprint of the vehicle and defines the size of $\boldsymbol{\mathcal{T}}$, and $r_{\boldsymbol{\Omega}} \geq r_{\boldsymbol{\mathcal{T}}}$ is a scalar defining the size of the avoidance region $\boldsymbol{\Omega}$. In [12, 13], the antitarget region is interpreted as the region where a collision may occur, and the avoidance region is a safety region not desirable to enter. During algorithm testing, any point inside $\boldsymbol{\mathcal{T}}$ is considered to be a collision.

As a result of the new regions, a new distance function $\rho'(u, r)$ is defined as:

$$\rho'(u, r) = max(\rho(u, r) - \Delta_s, 0), \tag{3.15}$$

where Δ_s is the distance the vehicle will travel until the next iteration of the algorithm, and $\rho(u, r)$ is the distance to reach $\boldsymbol{\mathcal{T}}$ along the following trajectory. A modified set of admissible velocities using $\rho'(u, r)$ is defined as:

$$V_a = \left\{ (u, r) \middle| u \leq \sqrt{2\rho'(u, r)|\dot{u}_{min}|} \wedge |r| \leq \begin{Bmatrix} \sqrt{2\rho'(u, r)|\dot{r}_{max}|}, & r < 0 \\ \sqrt{2\rho'(u, r)|\dot{r}_{min}|}, & r \geq 0 \end{Bmatrix} \right\}, \tag{3.16}$$

to take the asymmetrical set of possible yaw rates into account.

The modified objective function used in the search space is now defined as:

$$\max_{(u,r)} G(u, r) = \alpha \cdot yawrate(r, r_d') + \beta \cdot dist(u, r) + \gamma \cdot velocity(u, u_d'). \tag{3.17}$$

The yaw rate function $yawrate(r, r_d')$ replaces the heading function from the original algorithm. The heading function uses the desired heading ψ_d, while the

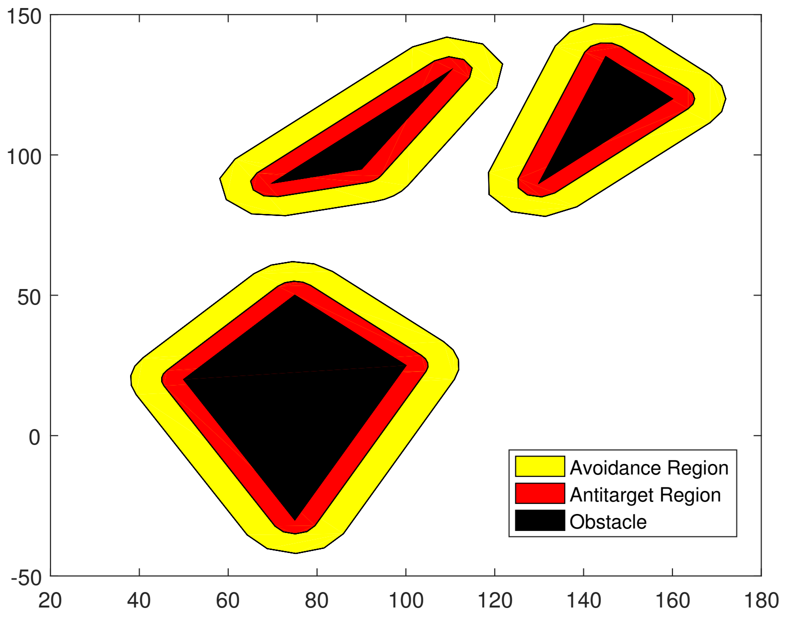


Figure 3.2: Avoidance region $\boldsymbol{\Omega}$ and antitarget region $\boldsymbol{\mathcal{T}}$.

yaw rate function uses a desired yaw rate r_d' to improve generality and flexibility [13]. The yaw rate function is given as:

$$yawrate(r, r_d') = 1 - \frac{|r_d' - r|}{\max\limits_{r \in V_r}(|r_d' - r|)}. \tag{3.18}$$

The distance function $dist(u, r)$ is now proportional to the approximated time until a collision occurs along the given trajectory, instead of the distance along it. The function is now taking the velocity in consideration, and is given as:

$$dist(u, r) = \frac{\bar{\rho}(u, r)}{\int_0^T \left\| \boldsymbol{\chi}(u, r, t) \right\|_2 dt}, \tag{3.19}$$

where $\bar{\rho}(u, r)$ is the distance to reach $\boldsymbol{\Omega}$ along the trajectory formed by the pair (u,r), and $\boldsymbol{\chi}(u, r, t)$ is the predicted surge and sway speed of the vehicle along the trajectory. The distance $\boldsymbol{\chi}(u, r, t)$ can be computed by the functions used for trajectory prediction (3.37) and (3.38):

$$\boldsymbol{\chi}(u, r, t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \bar{\boldsymbol{\nu}}(t | u_d, r_d), \tag{3.20}$$

where $\bar{\boldsymbol{\nu}}(y | u_d, r_d)$ is the solution of (3.37) and (3.38) at given time t.

The velocity function $velocity(u, u'_d)$ is now considering the difference between the surge speed, u and the desired surge speed u'_d. This makes the objective function favor velocities that are close to the desired surge speed. The function is given as:

$$velocity(u, u'_d) = 1 - \frac{|u'_d - u|}{\max\limits_{u \in V_r}(|u'_d - u|)}. \tag{3.21}$$

3.1.2 Modified Trajectory Prediction

A modified trajectory prediction has been introduced in [13, 12] to account for second order nonholonomic constraints. Due to the system being nonholonomic it will not be possible to linearize the system fully by feedback. In the proposed modification, linearization of the surge and yaw dynamics are done using partial feedback linearization, while the sway motion is left uncontrolled. To predict the trajectories, the closed loop dynamics are derived and used for simulation.

By putting $\dot{\boldsymbol{\nu}}$ in (2.10) by itself and inserting (2.12) the system kinetics can be expressed as

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f} - \boldsymbol{n}(\boldsymbol{\nu}), \tag{3.22}$$

where

$$\boldsymbol{n}(\boldsymbol{\nu}) = \boldsymbol{M}^{-1}(\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}) \tag{3.23}$$

is introduced to simplify notation. Two matrices $\boldsymbol{\Gamma}_1$ and $\boldsymbol{\Gamma}_2$ are introduced to separate the system:

$$\boldsymbol{\Gamma}_1 \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.24a}$$

$$\boldsymbol{\Gamma}_2 \triangleq \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \tag{3.24b}$$

which have the property $\boldsymbol{\Gamma}_1^\top\boldsymbol{\Gamma}_1 + \boldsymbol{\Gamma}_2^\top\boldsymbol{\Gamma}_2 = \boldsymbol{I}$. Consequently, (3.22) can be written as

$$\begin{aligned} \dot{\boldsymbol{\nu}} &= (\boldsymbol{\Gamma}_1^\top\boldsymbol{\Gamma}_1 + \boldsymbol{\Gamma}_2^\top\boldsymbol{\Gamma}_2)\boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f} - \boldsymbol{n}(\boldsymbol{\nu}) \\ &= \boldsymbol{\Gamma}_1^\top(\boldsymbol{\Gamma}_1\boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f} - \boldsymbol{\Gamma}_1\boldsymbol{n}(\boldsymbol{\nu})) - \boldsymbol{\Gamma}_2^\top\boldsymbol{\Gamma}_2\boldsymbol{n}(\boldsymbol{\nu})) \end{aligned}. \tag{3.25}$$

By using $\boldsymbol{\Gamma}_1$ to map the surge and yaw dynamics and $\boldsymbol{\Gamma}_2$ to map the sway dynamics, the system (2.10) is divided into two parts. The control law is chosen to be

$$\boldsymbol{f} = (\boldsymbol{\Gamma}_1\boldsymbol{M}^{-1}\boldsymbol{B})^{-1}(\boldsymbol{\Gamma}_1\boldsymbol{n}(\boldsymbol{\nu}) + \boldsymbol{a}_{1_d}), \tag{3.26}$$

where $\boldsymbol{a}_{1_d}$ is the desired acceleration, selected by using a proportional controller:

$$\begin{aligned} \boldsymbol{a}_{1_d} = \begin{bmatrix} \dot{u}_d \\ \dot{r}_d \end{bmatrix} &= \dot{\boldsymbol{\nu}}_{1_d} - \boldsymbol{K}_p(\boldsymbol{\nu}_1 - \boldsymbol{\nu}_{\boldsymbol{1_d}}) \\ &= \dot{\boldsymbol{\nu}}_{1_d} - \boldsymbol{K}_p(\boldsymbol{\Gamma}_1\boldsymbol{\nu} - \boldsymbol{\nu}_{\boldsymbol{1_d}}), \end{aligned} \tag{3.27}$$

where $\boldsymbol{\nu}_1 = \boldsymbol{\Gamma}_1\boldsymbol{\nu} = [u \;\; r]^\top$, $\boldsymbol{\nu}_{1_d} = [u_d \;\; r_d]^\top$, and $\boldsymbol{K}_p = \text{diag}(k_{p_u}, k_{p_r}) > 0$ is the gain matrix of the proportional controller. It is shown in [13] that $(\boldsymbol{\Gamma}_1\boldsymbol{M}^{-1}\boldsymbol{B})^{-1}$

always exists, by exploiting that $\boldsymbol{M}$ is positive definite and that the system is controllable in surge and yaw, to prove full rank. Inserting for $\boldsymbol{f}$ from (3.26) into (2.10) gives

$$\dot{\boldsymbol{\nu}} = \boldsymbol{\Gamma}_1^\top \boldsymbol{a}_{1_d} - \boldsymbol{\Gamma}_2^\top \boldsymbol{\Gamma}_2 \boldsymbol{n}(\boldsymbol{\nu}). \tag{3.28}$$

By defining

$$\tilde{\boldsymbol{\nu}} = \begin{bmatrix} \tilde{u} \\ v \\ \tilde{r} \end{bmatrix} \triangleq \boldsymbol{\nu} - \boldsymbol{\Gamma}_1^\top \boldsymbol{\nu}_{1_d}, \tag{3.29}$$

the dynamics in (3.28) can be expressed as

$$\dot{\tilde{\boldsymbol{\nu}}} = \begin{bmatrix} \dot{\tilde{u}} \\ \dot{v} \\ \dot{\tilde{r}} \end{bmatrix} = -\boldsymbol{\Gamma}_1^\top \boldsymbol{K}_p \boldsymbol{\Gamma}_1 \tilde{\boldsymbol{\nu}} - \boldsymbol{\Gamma}_2^\top \boldsymbol{\Gamma}_2 \boldsymbol{n}(\boldsymbol{\nu}). \tag{3.30}$$

From (3.30), the surge and yaw velocity is given as $\dot{\tilde{u}} = k_u \tilde{u}$ and $\dot{\tilde{r}} = -k_r \tilde{r}$. Hence, the selected $\boldsymbol{f}$ in (3.26) yields linearity in surge and yaw. The sway motion of the system is still nonlinear, expressed as $\dot{v} = -\boldsymbol{\Gamma}_2^\top \boldsymbol{\Gamma}_2 \boldsymbol{n}(\boldsymbol{\nu})$. To achieve linear dynamics in sway, an approximation of $\boldsymbol{n}(\boldsymbol{\nu})$ using first-order Taylor approximation is introduced as:

$$\begin{aligned} \boldsymbol{n}(\boldsymbol{\nu}) &\approx \boldsymbol{n}(\boldsymbol{\nu}*) + \left.\frac{\partial \boldsymbol{n}(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}}\right|_{\boldsymbol{\nu}=\boldsymbol{\nu}*} (\boldsymbol{\nu} - \boldsymbol{\nu}*) \\ &= \boldsymbol{n}(\boldsymbol{\nu}*) + \boldsymbol{N}\boldsymbol{\nu} - \boldsymbol{N}\boldsymbol{\nu}* \\ &= \boldsymbol{N}\boldsymbol{\nu} + \boldsymbol{b}(\boldsymbol{\nu}*), \end{aligned} \tag{3.31}$$

where

$$\boldsymbol{N} = \left.\frac{\partial \boldsymbol{n}(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}}\right|_{\boldsymbol{\nu}=\boldsymbol{\nu}*}, \tag{3.32}$$

is the Jacobian matrix of $\boldsymbol{n}(\boldsymbol{\nu})$, further described in [13], $\boldsymbol{\nu}*$ is the current velocity, used as linearization point, and

$$\boldsymbol{b}(\boldsymbol{\nu}*) = \boldsymbol{n}(\boldsymbol{\nu}) - \boldsymbol{N}\boldsymbol{\nu}*. \tag{3.33}$$

By defining

$$\begin{aligned} \boldsymbol{A} &= -(\boldsymbol{\Gamma}_1^\top \boldsymbol{K}_p \boldsymbol{\Gamma}_1 + \boldsymbol{\Gamma}_2^\top \boldsymbol{\Gamma}_2 \boldsymbol{N}) \\ \boldsymbol{\beta} &= -\boldsymbol{\Gamma}_2^\top \boldsymbol{\Gamma}_2 \boldsymbol{N} \boldsymbol{\Gamma}_1^\top \\ \boldsymbol{G} &= -\boldsymbol{\Gamma}_2^\top \boldsymbol{\Gamma}_2 \boldsymbol{b}(\boldsymbol{\nu}*), \end{aligned} \tag{3.34}$$

the dynamics in (3.30) can be expressed as:

$$\dot{\tilde{\boldsymbol{\nu}}} = \boldsymbol{A}\tilde{\boldsymbol{\nu}} + \boldsymbol{\beta}\boldsymbol{\nu}_{1_d} + \boldsymbol{G}, \tag{3.35}$$

and the solution is

$$\tilde{\boldsymbol{\nu}}(t) = e^{\boldsymbol{A}t}\tilde{\boldsymbol{\nu}}(t_0) + \int_{t_0}^{t} e^{\boldsymbol{A}(t-\sigma)}(\boldsymbol{\beta}\boldsymbol{\nu}_{1_d}(\sigma) + \boldsymbol{G})\mathrm{d}\sigma. \tag{3.36}$$

If the initial time t_0 is set to zero, (3.36) can be expressed as [20]:

$$\tilde{\boldsymbol{\nu}}(t) = e^{\boldsymbol{A}t}\tilde{\boldsymbol{\nu}}(0) - \boldsymbol{A}^{-1}(\boldsymbol{I} - e^{\boldsymbol{A}t})(\boldsymbol{\beta}\boldsymbol{\nu}_{1_d} + \boldsymbol{G}). \tag{3.37}$$

Lastly, the predicted trajectory is found by simulating the kinematics of the system (2.9). The simulations are done numerically using modified Euler method, which gives

$$\begin{aligned} \boldsymbol{\eta}(t_{n+1}) &= \boldsymbol{\eta}(t_n) + h\boldsymbol{k}_2 \\ \boldsymbol{k}_1 &= \boldsymbol{R}(\boldsymbol{\eta}(t_n))\boldsymbol{\nu}(t_n) \\ \boldsymbol{k}_2 &= \boldsymbol{R}(\boldsymbol{\eta}(t_n) + \frac{h}{2}\boldsymbol{k}_1)\boldsymbol{\nu}(t_n + \frac{h}{2}), \end{aligned} \tag{3.38}$$

where h is the time step of the integration.

The approach and deriving of the predicted trajectories are based on Eriksen's approach in [12, 13]. However, a different vessel model is used for the simulations in Chapter 6, which might yield different results. Especially, the nonlinear damping of the ASV model may reduce the accuracy of the linearization in (3.31).

To reduce the inaccuracy caused by the nonlinear damping matrix, we define a new trajectory prediction, which updates the linearization point to $\boldsymbol{\nu}* = \boldsymbol{\nu}(t_n)$ at every iteration of the modified Euler method. Accordingly, (3.32), (3.33) and (3.34) are updated. The frequent matrix updates cause an increase in the calculation cost when generating trajectory predictions. Originally, the matrices are only calculated once every DW iteration. Now, however, the matrices are calculated numerous times for every velocity pair in the search space. Note that the system which used to be linear time-invariant system now is a linear time-varying system.

3.2 Algorithm B

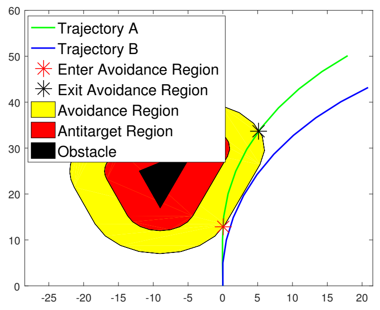
Algorithm B is based on Algorithm A but differs in the distance function. A modification to the distance function is introduced to motivate exiting the avoidance region. In addition to considering the time until entering the avoidance region, the new distance function $dist_B(u, r)$ considers the time until reaching the avoidance region exit along the trajectory. If a trajectory is close to enter or inside the avoidance region, the algorithm should weight the trajectory based on how close the exit of the avoidance region is. If the trajectory is not immediately leading to the avoidance region, it should be weighted by the time until entering the avoidance region, as in Algorithm A. The distance to the closest avoidance region along the trajectory is given by the function $\bar{\rho}(u, r)$, while the distance to exit the avoidance region is given by the function $\rho_B(u, r)$ expressed as:

$$\rho_B = \int_0^T \left\|\boldsymbol{\chi}(u, r, t)\right\|_2 dt - \rho_{\boldsymbol{\Omega}}(u, r), \tag{3.39}$$

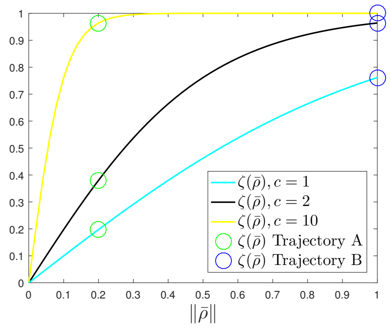
where $\boldsymbol{\chi}(u,r,t)$ is the ASV velocity along the predicted trajectory given in (3.20) and $\rho_{\boldsymbol{\Omega}}(u,r)$ is the distance until the trajectory exits the avoidance region $\boldsymbol{\Omega}$. Reaching the antitarget region $\boldsymbol{\mathcal{T}}$ before exiting the avoidance region $\boldsymbol{\Omega}$ along a predicted trajectory yields $\rho_B(u,r) = 0$. In Figure 3.3a, $\bar{\rho}(u,r)$ and $\rho_B(u,r)$ are given by the distance to the marked points where Trajectory A enters and exits the avoidance region, respectively. To be able to smoothly switch the weighting between $\bar{\rho}(u,r)$ and $\rho_B(u,r)$, a switch $\zeta(\bar{\rho})$ using the hyperbolic tangent function $tanh(\cdot)$ is introduced as:

$$\zeta(\bar{\rho}) = tanh(c\|\bar{\rho}\|) \in [0,1], \tag{3.40}$$

where $c > 0$ is a tuning constant deciding how smooth the switching will be, as presented in Figure 3.3b.



(a) Trajectory examples A and B.



(b) Values of $\zeta(\bar{\rho})$ for trajectories A and B for different values of tuning parameter c, with $\|\bar{\rho}\| = 0.2$ and $\|\bar{\rho}\| = 1$ for trajectories A and B, respectively.

Figure 3.3: Algorithm B examples showing how different values of the tuning parameter c changes the weighting in $dist_B(u,r)$.

Finally, a new distance function can be expressed as:

$$dist_B(u,r) = \frac{\zeta(\bar{\rho}) \cdot \bar{\rho}(u,r) + (1-\zeta(\bar{\rho})) \cdot \rho_B(u,r)}{\int_0^T \left\|\boldsymbol{\chi}(u,r,t)\right\|_2 dt} \in [0,1]. \tag{3.41}$$

The new distance function leads to a new cost function expressed as:

$$\begin{aligned} &\max_{(u,r)} G(u,r) = \alpha \cdot yawrate(r, r'_d) + \beta \cdot dist_B(u,r) + \gamma \cdot vel(u, u'_d) \\ &s.t.\ (u,r) \in V_r, \end{aligned} \tag{3.42}$$

where $yawrate(r, r'_d)$ and $vel(u, u'_d)$ are the same functions used in the cost function (3.17) in Algorithm A.

By choosing a significantly large c in (3.40) Algorithm B will act identically to Algorithm A when the vehicle is outside the avoidance region. If the ASV resides inside the avoidance region, however, the algorithm will be able to choose a velocity pair that leads the ASV out of the region. Having a lower c allows the algorithm to consider whether trajectories leads through the avoidance region or towards an obstacle before the ASV is inside the avoidance region.

3.3 Algorithm C

Algorithm B introduces a way for the DW algorithm to exit the avoidance region once entered. However, the algorithm does not consider where the trajectory leads past the entrance of the avoidance region unless the distance function $\bar{\rho}(u, r)$ is small. Furthermore, once the vehicle is inside the avoidance region the algorithm does not consider where the trajectory leads past the exit point. To consider the complete predicted trajectory, we introduce a new function $\rho_C(u, r)$, which returns the portion of the predicted trajectory that resides outside the avoidance region. By discretizing the predicted trajectory into $N > 0$ parts the new function $\rho_C(u, r)$ is expressed as:

$$\rho_C(u, r) = \frac{\sum_{n=1}^{N} \frac{\lambda(u,r,n)}{\sqrt{n}}}{\sum_{n=1}^{N} \frac{1}{\sqrt{n}}} \in [0, 1], \tag{3.43}$$

where

$$\lambda(u, r, n) = \begin{cases} 0, & \text{if part } n \text{ of the trajectory resides inside } \mathbf{\Omega} \\ 1, & \text{otherwise} \end{cases}. \tag{3.44}$$

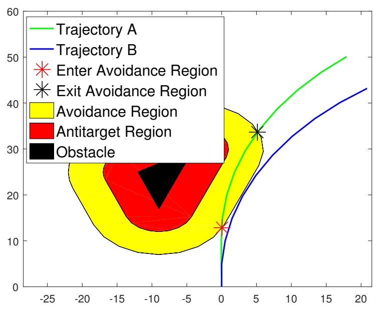
Figure 3.4 presents two examples of predicted trajectories, and how the summation in (3.43) grows along them.

To prevent Algorithm C from choosing trajectories cutting through the avoidance region unnecessarily, $\bar{\rho}(u, r)$ is combined with $\rho_C(u, r)$ and form a new distance function $dist_C(u, r)$ defined as:

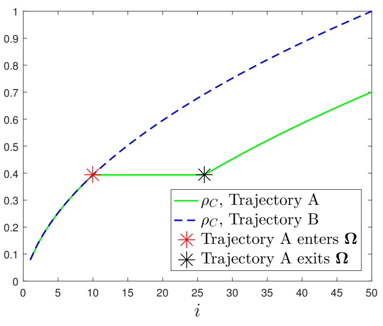
$$dist_C(u, r) = \kappa \frac{\bar{\rho}(u, r)}{\int_0^T \left\| \boldsymbol{\chi}(u, r, t) \right\|_2 dt} + (1 - \kappa)\rho_C(u, r) \in [0, 1], \tag{3.45}$$

where $\boldsymbol{\chi}(u, r, t)$ is the predicted velocity (3.20), and $\kappa \in [0, 1]$ is a tuning parameter deciding the weight between the distant parts $\bar{\rho}(u, r)$ and $\rho_C(u, r)$. The new distance function leads to a new cost function:

$$\begin{aligned} &\max_{(u,r)} G(u, r) = \alpha \cdot yawrate(r, r'_d) + \beta \cdot dist_C(u, r) + \gamma \cdot vel(u, u'_d) \\ &s.t.\ (u, r) \in V_r \end{aligned}, \tag{3.46}$$



(a) Trajectory examples A and B.



(b) The scaled summation from $n = 1$ to $n = i$ in (3.43) along trajectories A and B for $N = 50$.

Figure 3.4: Algorithm C examples showing how (3.44) differs when the trajectory is inside or outside of the avoidance region.

where $yawrate(r, r'_d)$ and $vel(u, u'_d)$ are the same functions used in the cost function (3.17) in Algorithm A.

The distance function $dist_C(u, r)$ enables the algorithm to consider what lays beyond the entrance of the avoidance region and whether a trajectory leads out of the region. Consequently, the algorithm is less hesitant to enter the avoidance region, compared to algorithms A and B, if the predicted trajectory leads through the region.

Chapter 4

Hybrid COLAV Method Using the DW Algorithm

The algorithms in Chapter 3 return promising results for the DW algorithm as a reactive COLAV method. The algorithm is, however, a local method and there is a significant risk that the ASV can be stuck in a local minima, which is a sufficient motivation to use hybrid COLAV methods. Furthermore, hybrid methods allows using deliberate algorithms and has other advantages as described in Section 2.6.

The proposed hybrid COLAV method consists of a global method (RRT) yielding a planned trajectory, and a Hybrid Dynamic Window (HDW) algorithm being guided by the predefined trajectory.

4.1 Interface Between DW Algorithm and Deliberate COLAV

To combine a deliberate and a reactive COLAV method, an interface between the methods is necessary. Considering that the deliberate method is intended to guide the reactive method, the HDW algorithm has to be motivated to follow the planned path or trajectory. In algorithms A, B and C, the desired yaw rate r'_d is chosen based on LOS guidance, and the desired surge speed u'_d is set to a constant value. Following the desired velocity will cause the ASV to smoothly converge towards the LOS path. However, it may be desirable when guiding ASVs to allow for curved paths or trajectories from the deliberate method, hence a LOS based guidance method does not hold. Note that by using deliberate methods which generate trajectories, moving obstacles can be avoided.

Instead of having a straight path between two waypoints, we now have a curved trajectory which contains information about the complete planned vessel state $\boldsymbol{p}_{pp}(t)$ at any time $t \geq 0$. The planned vessel state $\boldsymbol{p}_{pp}(t)$ describes the full pose and velocity of the vessel along the trajectory as:

$$\boldsymbol{p}_{pp}(t) = [x_{pp}(t), y_{pp}(t), \psi_{pp}(t), u_{pp}(t), v_{pp}(t), r_{pp}(t)]^\top . \tag{4.1}$$

By applying an interface between the reactive method and deliberate method, the ASV can be guided towards the trajectory using a trajectory tracking method.

4.2 Hybrid Dynamic Window (HDW) Algorithm

The HDW algorithm is intended to select velocity pairs that follow the planned trajectory. However, a single velocity pair may not yield a fitting trajectory since the desired turn rate may change along the planned trajectory. To account for this, trajectories are defined using multiple velocity pairs (2 or more). The trajectories are generated by first making predictions based on single velocity pair within the search space as in the original DW algorithm. By using the end state of every predicted trajectory from the original search space, a new search space is generated in the same way. For each new search space, new trajectory predictions are generated based on the velocity pairs within it. This enables the algorithm to consider trajectories where the desired surge speed and yaw rate changes along the trajectory. Figure 4.1 illustrates the use of double velocity pairs, where the first search space contains three velocity pairs, while the new search space at the end of the trajectory predictions consists of five different velocity pairs each. A related modification is implemented for the DW algorithm in [29]. Lastly, the HDW algorithm will choose the optimal trajectory based on a cost function. Note that by using multiple velocity pairs to define trajectories, the computation load increases exponentially with the numbers of velocity pairs.

To be able to follow the planned trajectory, the cost function has to be motivated to do so. By using an external trajectory tracker, the cost functions in DW algorithms A, B and C can be applied directly in the hybrid COLAV method. However, by modifying the DW algorithm, it can be used directly as a trajectory tracker, which will make an external guidance system redundant. Two different cost function options are proposed for use in the HDW algorithm.

4.2.1 HDW Option 1 - External Trajectory Tracker

The first HDW option has a similar structure as algorithms A, B and C. It differs by having varying desired surge speed u_d' and yaw rate r_d' based on a planned trajectory. In addition, the predicted trajectories are defined by multiple velocity pairs.

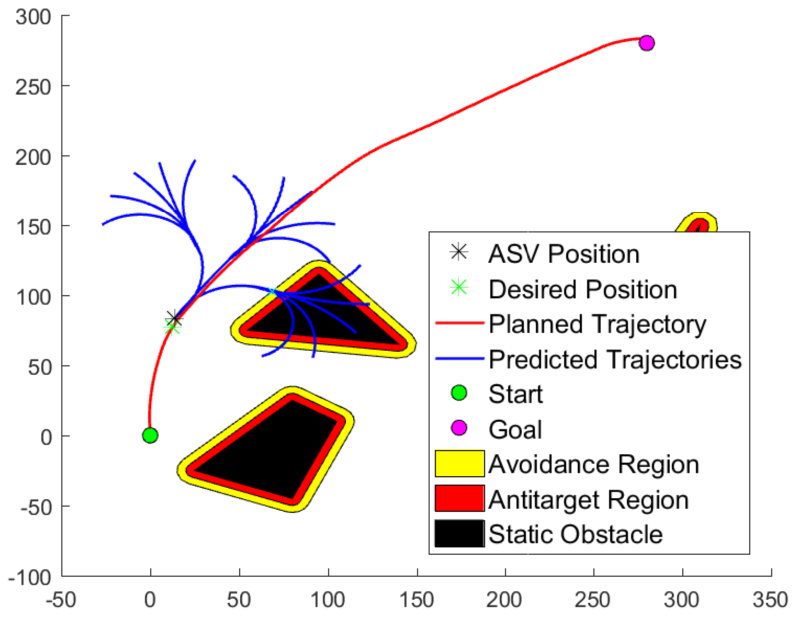


Figure 4.1: HDW algorithm using double velocity pairs following a planned trajectory.

A method for tracking curved paths is described in Section 2.2.3. The method returns a desired surge speed u_d and heading ψ_d based on the difference between the ASV state $\boldsymbol{p}(\theta)$ and the planned state $\boldsymbol{p}_{pp}(\theta)$, where θ is a controllable path parameter. Since the deliberate methods in the hybrid COLAV method will return trajectories, the path parameter is set to be equal to the time t. The desired heading from (2.48) in the tracking method is then used in a controller to find the desired yaw rate r'_d for the DW algorithm. The control law is defined as

$$r'_d = -k_\psi(\psi - \psi_d) + \dot{\psi}_d, \tag{4.2}$$

where $k_\psi > 0$ is a constant gain. Since the path parameter is given by time, the path tangential speed U_{PP} is equal to the planned speed. By rearranging (2.42), the desired speed can be defined as:

$$u'_d = \frac{U_{PP} - \gamma s}{cos(\chi - \chi_t)}. \tag{4.3}$$

Note that the desired speed has a singularity if the ASV course is perpendicular to the planned trajectory course.

Finally, the desired velocity pair (u'_d, r'_d) is used in the HDW algorithm cost function as:

$$\begin{aligned} &\max_{(u,r)} G(u,r) = \alpha \cdot yawrate(r, r'_d) + \beta \cdot dist(u,r) + \gamma \cdot vel(u, u'_d) \\ &s.t.\ (\boldsymbol{u}, \boldsymbol{r}) \in \bar{\boldsymbol{V}}_r \end{aligned}, \tag{4.4}$$

where $dist(\boldsymbol{u}, \boldsymbol{r})$ can be either one of the distance functions in DW algorithm A, B or C, and $yawrate(r, r'_d)$ and $vel(u, u'_d)$ are defined in (3.18) and (3.21), respectively. The search space $\bar{\boldsymbol{V}}_r \in \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ contains a sub search space for each of the velocity pairs in $(\boldsymbol{u}, \boldsymbol{r})$, where $(\boldsymbol{u}, \boldsymbol{r})$ are sets of multiple velocity pairs defined as:

$$\begin{aligned} \boldsymbol{u} &= \{u_1, \dots, u_{\bar{K}}\} \\ \boldsymbol{r} &= \{r_1, \dots, r_{\bar{K}}\} \end{aligned}, \tag{4.5}$$

where $\bar{K}$ is the amount of velocity pair used to form each predicted trajectory in HDW.

If the velocity pair (u'_d, r'_d) is within the range of the search space, the pair is added to it so that the algorithm can return the optimal trajectory.

Figure 4.2 presents an overview of the architecture of HDW option 1. The figure shows how the trajectory tracker uses the ASV state and planned trajectory to find the desired surge speed u'_d and heading ψ_d.

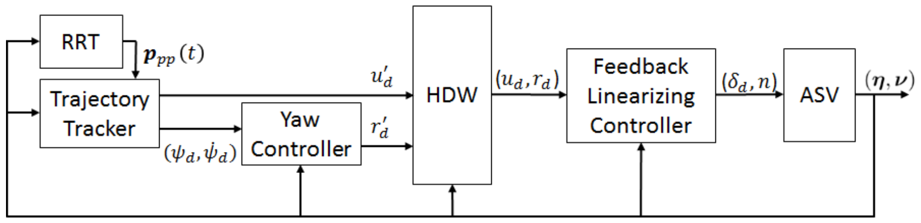


Figure 4.2: Architecture of HDW option 1.

4.2.2 HDW Option 2 - Trajectory Alignment

The original DW algorithm and algorithms A, B and C are all dependent on an external guidance system to follow a desired path. This is due to the functions $vel(u, r)$ and $heading(u, r)$ in the original DW algorithm, and $vel(u, u'_d)$ and $yawrate(r, r'_d)$ for algorithms A, B and C. Specifically, the functions need the desired velocity from the guidance system as input. By replacing these functions with a function that motivates the ASV to travel along the desired trajectory, the need of an external guidance system is removed.

To motivate the DW algorithm to return velocity pairs leading the ASV along the planned trajectory, a new function is introduced in the cost function. The function returns a value based on how well the predicted trajectory of a velocity pair aligns with the planned trajectory. The predicted trajectory of a set of velocity pairs yields a complete vessel state prediction $\boldsymbol{p}_{pt}(t)$ at time t along the trajectory defined as:

$$\boldsymbol{p}_{pt}(t) = [x_{pt}(t), y_{pt}(t), \psi_{pt}(t), u_{pt}(t), v_{pt}(t), r_{pt}(t)]^\top. \tag{4.6}$$

By using both the state $\boldsymbol{p}_{pp}(t)$ along the planned trajectory and the predicted state $\boldsymbol{p}_{pt}(t)$ along the predicted trajectory, a comparison can be performed between the two curves. The curves are discretized into $N > 0$ points between the current time t and $t+t_{pt}$, where t_{pt} is the time frame of the predicted trajectory. Note that t_{pt} imparts how far ahead in time the alignment function compares the trajectories. Finally, a trajectory alignment function is defined as:

$$align(\boldsymbol{p}_{pt}, \boldsymbol{p}_{pp}) = \frac{k_a}{N} \sum_{i=1}^{N} d(\boldsymbol{p}_{pt}(t_i), \boldsymbol{p}_{pp}(t_i)), \tag{4.7}$$

where t_i is expressed as:

$$t_i = t + \frac{i \cdot t_{pt}}{N}, \tag{4.8}$$

$k_a = 1$ 1/m to make the align function unitless, and $d(\boldsymbol{p}_{pt}(t_i), \boldsymbol{p}_{pp}(t_i))$ is the Euclidean distance between planned and predicted state at time t_i expressed as:

$$d(\boldsymbol{p}_{pt}(t_i), \boldsymbol{p}_{pp}(t_i)) = \left\| \boldsymbol{p}_{pp}(t_i) - \boldsymbol{p}_{pt}(t_i) \right\|_2. \tag{4.9}$$

Now, the algorithm favors velocity pairs that will lead the ASV along the planned trajectory in the near future. Unlike HDW option 1, the algorithm can take upcoming turns into account in the cost function. Figure 4.3 illustrates the distance at each value of t_i in the trajectory alignment function.

Any of the distance functions (3.19), (3.41) and (3.45) in algorithms A, B and C, respectively, can be chosen for implementing $dist(\boldsymbol{u}, \boldsymbol{r})$. Note that the distance function use the set of multiple velocity pairs $(\boldsymbol{u}, \boldsymbol{r})$ defined in (4.5). By introducing a single tuning parameter $\bar{\alpha} \in [0, 1]$, a new cost function for the HDW algorithm can be defined as:

$$\begin{aligned} &\max_{(u,r)} G(u,r) = \bar{\alpha} \cdot dist(\boldsymbol{u}, \boldsymbol{r}) - (1 - \bar{\alpha}) \cdot align(\boldsymbol{p}_{pt}, \boldsymbol{p}_{pp}) \\ &s.t. \ (\boldsymbol{u}, \boldsymbol{r}) \in \bar{\boldsymbol{V}}_r \end{aligned}, \tag{4.10}$$

where $\bar{\boldsymbol{V}}_r \in \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ contains a search space for each of the velocity pairs in $(\boldsymbol{u}, \boldsymbol{r})$. The tuning parameter $\bar{\alpha}$ sets the weighting between the obstacle distance and the trajectory alignment in the cost function. Specifically, $\bar{\alpha}$ sets how risk averse the ASV is when tracking the trajectory. By increasing $\bar{\alpha}$ towards 1, the HDW algorithm will favor avoiding the avoidance region over following the

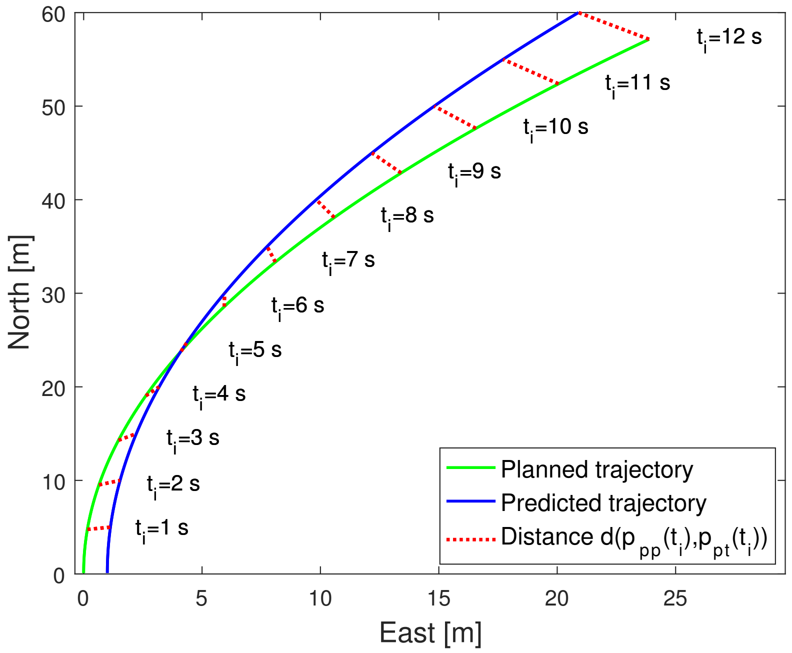


Figure 4.3: Trajectory alignment for $N = 12$, where the trajectory prediction time frame $t_{pt} = 12$ s. The RRT algorithm generates the planned trajectory.

planned trajectory. If $\bar{\alpha} = 1$, the algorithm cares only about avoiding obstacles and has no desire to follow the trajectory or reach the goal. If $\bar{\alpha} = 0$ on the other hand, the algorithm will only care about following the planned trajectory but will still only consider admissible velocity pairs.

To optimize the search space, the desired velocity pair from the external trajectory tracker presented for HDW option 1 is added. Consequently, the algorithm has the option to always choose (u'_d, r'_d) if the pair is within the search space limits. An architecture overview is presented in Figure 4.4, where the optional trajectory tracker resides inside the HDW algorithm. Unlike in HDW option 1, the HDW algorithm takes in the complete planned trajectory directly.

When using multiple velocity pairs in the trajectory predictions, there is a chance that first part of the trajectory aligns perfectly with the planned trajectory, while the last part does not. To avoid not choosing the perfect velocity pair, it is possible to only use the first trajectory part in the align function. Note that how far ahead in time the trajectory prediction covers decides the lookahead time of the align function. Hence, by only using the part of the predicted trajectory from the first velocity pair, the lookahead time is significantly

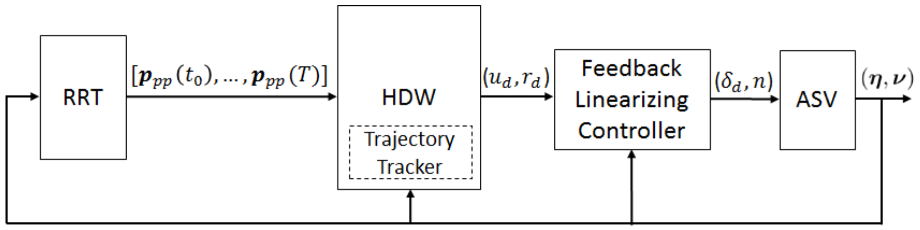

RRT
$[\boldsymbol{p}_{pp}(t_0), \dots, \boldsymbol{p}_{pp}(T)]$
HDW
Trajectory Tracker
(u_d, r_d)
Feedback Linearizing Controller
(δ_d, n)
ASV
$(\boldsymbol{\eta}, \boldsymbol{\nu})$

Figure 4.4: Architecture of HDW option 2.

reduced. If the trajectory prediction of the first velocity pair has a sufficiently long lookahead time, it is reasonable to only use the first velocity pair for the align function.

The HDW option 2 algorithm enables the use of any deliberate method that generates a trajectory. Unlike HDW option 1, option 2 is unable to follow time-invariant paths.

4.2.3 Comparison of HDW Options

Even though both the HDW options are suitable to apply in a hybrid COLAV method, only one will be used for detailed testing of the method. Therefore, a comparison is presented between the two.

The guidance of HDW option 1 is similar to the LOS guidance and can easily be applied to previous implemented DW algorithms. For a planned trajectory with few sudden turns, this option will smoothly converge to and stay on the trajectory. The trajectory tracking method does, however, only take the current planned state into account. Consequently, HDW option 1 can not react to upcoming turns ahead of time. An overview of the pros and cons of HDW option 1 is presented in Table 4.1.

HDW option 2 compares the predicted trajectory of every velocity pair with the planned state, and will therefore choose velocity pair based on how well the resulting trajectory aligns with the planned trajectory. Furthermore, option 2 can receive suggestions from an external trajectory tracker but is not dependent on it. Unlike the cost function of HDW option 1 which has three tuning parameters, the cost function of option 2 consists of only one. An overview of the pros and cons of HDW option 2 is presented in Table 4.2.

To evaluate how the HDW options perform, simulation results of the methods tracking a planned trajectory past obstacles are presented in Figure 4.5. The figure shows how HDW option 1 leads the ASV smoothly towards the planned trajectory when the change of yaw rate of the planned state is low. When the change of yaw rate is higher, the ASV overshoots more and uses some time

Table 4.1: Pros and cons for HDW option 1 - External trajectory tracker.

Pros	**Cons**
Similar to previous tested DW algorithms	Only consider planned state at current time
Search space contain optimal solution	Dependent on a good external trajectory tracker
Easily applicable with DW algorithms	Numerous tuning parameters
Can easily be adjusted to fit path tracking	

Table 4.2: Pros and cons for HDW option 2 - Trajectory alignment.

Pros	**Cons**
Few tuning parameters	May cut turns some
Search space can get input from external trajectory tracker	Dependent on a good trajectory prediction
Trajectory prediction is already implemented for DW	Can not be applied as path tracker
Takes planned state in near future into account	
Not dependent on external trajectory tracker	

to reach back to the planned trajectory. HDW option 2 leads the ASV more smoothly through the scenario but cuts the turns some compared to the planned trajectory. Further details about simulations and testing are described in chapters 5 and 6.

To sum up, option 2 follows the trajectory more smoothly and has a clear advantage compared to option 1 by only having one tuning parameter. Furthermore, option 2 differs more from previous tested DW algorithms and will therefore be of more interest for further testing. Consequently, the HDW algorithm will be based on option 2 in the final simulation results in Section 6.2.2.

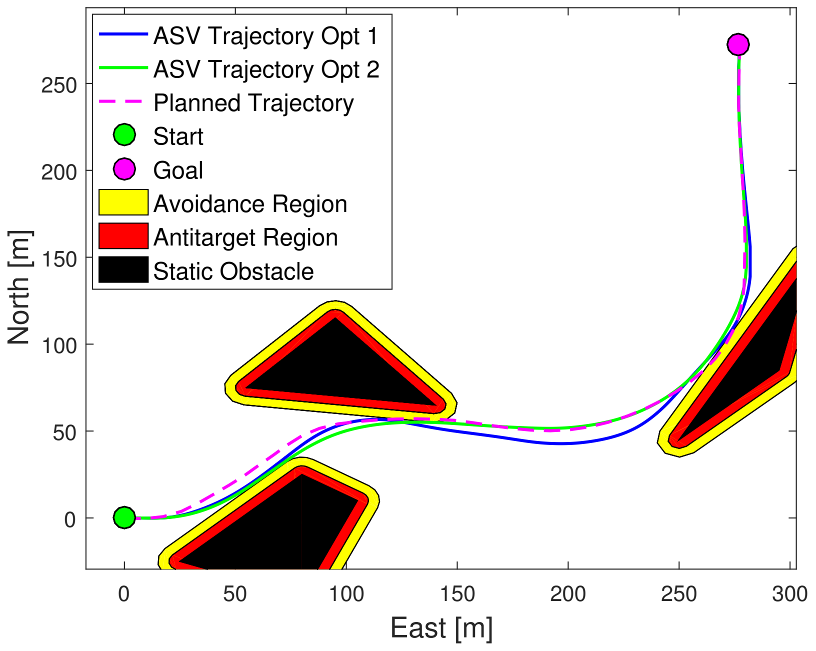


Figure 4.5: Comparison of HDW options.

Chapter 5

Simulator Implementation

This chapter describes the implementation of the DW algorithms, the RRT algorithm and the hybrid COLAV method. In addition, the chapter introduces the ASV model used for simulations in Chapter 6.

5.1 Simulator Model

The vessel model in the simulator is based on the simplifications done in Section 2.1.3, and by the approximations done by Loe in [28]. Some parameters of the vessel depicted in Figure 5.1 are given by [38], while most of them are estimated properties of Viknes 830, listed in Table 5.1.

Figure 5.1: Viknes 830 [38].

Table 5.1: Approximated parameters of Viknes 830.

Parameter	**Value**
Length	8.52 m
Width	2.97 m
Draugth	0.82 m
m	3980 kg
I_z	19703 kg/m^2
X_u	-50 kg/s
$X_{\|u\|u}$	-135 kg/m
X_{uuu}	0 kg s/m
Y_v	-200 kg/s
$Y_{\|v\|v}$	-2000 kg/m
Y_{vvv}	0 kg s/m
N_r	-1281 kg m/s
$N_{\|r\|r}$	0 kg m
N_{rrr}	-3224 kg m s
$F_{X,max}$	13100 N
$F_{X,min}$	-6550 N
$F_{N,max}$	2580 N m
u_{max}	10.5 m/s
$\delta_{\psi,max}$	15 °
$\dot{\delta}_{\psi,max}$	15 °/s
l_r	4 m
K_{δ_ψ}	98.55 kg/m^2 rad

The added mass $\boldsymbol{M}_a$ of the vessel is not defined, hence the added mass and Coriolis is set to zero:

$$\boldsymbol{M}_A = 0 \tag{5.1a}$$

$$\boldsymbol{C}_A = 0. \tag{5.1b}$$

It is argued in [28] that neglecting the added mass is an acceptable approximation. In addition, the mass distribution of the vessel is not identified. Hence, the inertia matrix $\boldsymbol{M}$ is simplified as

$$\boldsymbol{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix}. \tag{5.2}$$

The Coriolis and centripetal matrix in (2.20) can be a simplified to

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -mv \\ 0 & 0 & mu \\ mv & -mu & 0 \end{bmatrix}. \tag{5.3}$$

Since several of the estimated parameters in the damping matrix $\boldsymbol{D(\nu)}$ is zero, the damping matrix in (2.21) may now be written as

$$\boldsymbol{D(\nu)} = \begin{bmatrix} X_u + X_{|u|u}|u| & 0 & 0 \\ 0 & Y_v + Y_{|v|v}|v| & 0 \\ 0 & 0 & N_r + N_{rrr}r^2 \end{bmatrix}. \tag{5.4}$$

The force X is given by the actuation from the propeller and the moment N is the rudder moment. The lack of a propeller and rudder model in [28] causes the yaw moment of the rudder to be independent of surge speed. Assuming independence between yaw moment and surge speed is a very rough approximation, and is differing from the realistic model. Hence, a rudder model is needed. To include the rudder yaw moment dependency on the surge velocity, we propose to include a rudder model as described in (2.15). By using the parameters listed in Table 5.1 in (2.15), the rudder coefficient for Viknes 830 can be approximated as:

$$K_{\delta_\psi} = \frac{F_{N,max}}{u_{max}^2 l_r \delta_{\psi,max}}. \tag{5.5}$$

A realistic model of the propeller is dependent on the surge speed, but this has low impact for the collision avoidance simulation. Hence, the moment in surge is defined as:

$$X \in [F_{X,min}, F_{X,max}], \tag{5.6}$$

where X can change instantly, hence $X = F_X$.

The controllers of the system are selected as described in (3.26) to cancel out the nonlinear components in surge and yaw. Hence, the force and moment controller laws are given as

$$F_X = (X_u + X_{|u|u}|u|\, u) + m(-rv + K_{p_u}(u_d - u)) \tag{5.7a}$$

$$F_N = (N_r + N_{rrr}r^3) + I_z K_{p_r}(r_d - r), \tag{5.7b}$$

where the gains K_{p_u} and K_{p_r} are positive constants given in Table 5.2. A desired rudder deflection δ_{ψ_d} is found by inserting the yaw moment controller F_N into (2.15). The desired rudder is then used as reference for a low level proportional rudder controller. The rudder angle and angle rate is saturated by $\delta_{\psi,max}$ and $\dot{\delta}_{\psi,max}$, respectively. The maximum rudder angular rate is chosen to be $\dot{\delta}_{\psi,max} = 15$ °/s, to make a fairly realistic model. Finally, the forces from the rudder in the sway direction is given by

$$Y = l_r N. \tag{5.8}$$

The model used for predicting trajectories in DW algorithms A, B and C is found by solving the Jacobian matrix $\boldsymbol{N}$ from (3.32) and use this with $\boldsymbol{n}(\boldsymbol{\nu})$ defined in (3.23) to solve for $\boldsymbol{b}(\boldsymbol{\nu})$ in (3.33). The Jacobian matrix $\boldsymbol{N}$ is found to be

$$\boldsymbol{N} = \begin{bmatrix} (X_u + 2X_{|u|u}|u|)\frac{1}{m} & -r & -v \\ r & (Y_v + 2Y_{|v|v}|v|)\frac{1}{m} & u \\ 0 & 0 & (N_r + 3N_{rrr}r^2)\frac{1}{I_z} \end{bmatrix}, \tag{5.9}$$

and $\boldsymbol{n}(\boldsymbol{\nu})$ is found by solving (3.23), using the simulation model of Coriolis and damping from (5.3) and (5.4), respectively.

5.2 Algorithm Implementations

This section gives a description of the algorithm implementations, where a step wise summary is given for each algorithm.

5.2.1 DW Algorithms

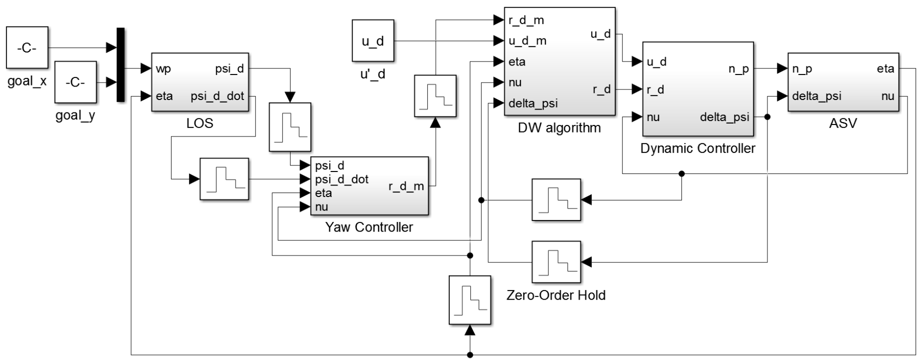


Figure 5.2: Architecture of DW algorithms A, B and C implemented in MATLAB Simulink.

The DW algorithm architecture presented in Figure 5.2 consists of a LOS guidance block, a yaw controller, a DW block, a controller, and an ASV block. The LOS block returns the desired heading ψ_d based on current state and goal position. The yaw controller uses ψ_d and the ASV state to generate the desired yaw rate r'_d, which is utilized by the DW algorithm. Further, the DW algorithm uses the ASV state, and the desired velocity pair (u'_d, r'_d) to select a velocity pair (u_d, r_d) for the controller. Finally, the rudder and propeller are controlled by the dynamic controller. The Zero-Order Hold blocks set the iteration frequency of the DW algorithm. A detailed description of the DW iteration is presented in Algorithm 2.

Possible Velocity

To find the possible velocity V_s in (3.10) for the search space, the stationary solutions of the ASV model (2.9) are approximated for surge speed u, yaw rate r and sway speed v. The stationary yaw rate can be expressed as:

$$N_{rrr}r^3 + N_r r + K_{\delta_\psi} u^2 l_r \delta_{\psi,max} = 0. \tag{5.10}$$

Algorithm 2: DW iteration

```
1:  $r_{max} \leftarrow$ found by solving for $r_{max}$ in (5.10) using current $u$
2:  $[\dot{u}_{min}, \dot{u}_{max}, \dot{r}_{min}, \dot{r}_{max}] \leftarrow$ found by solving (3.6)
3:  $V_s \leftarrow [0, u_{max}], [-r_{max}, r_{max}]$ (2.55)
4:  $V_d \leftarrow [u + \dot{u}_{min}T, u + \dot{u}_{max}T], [r + \dot{r}_{min}T, r + \dot{r}_{max}T]$ (2.54)
5:  $V \leftarrow$ **union**$(V_s, V_d)$
6:  **for** $i = 1$ **to** $n$ **do**
7:      **for** $j = 1$ **to** $m$ **do**
8:          disc_$V((i-1)n + j) \leftarrow$ velocity pair $[u(i), r(i)]$ in grid$(V)$
9:  **for** $i = 1$ **to** $n \cdot m$ **do**
10:     $[u, r] \leftarrow$disc_$V(i)$
11:     **for** t = 1 **to** prediction_time/timestep **do**
12:         [x($t$),y($t$),$\psi(t)$] $\leftarrow$ predicted trajectory for $[u, r]$ using (3.38)
13:     admissible($i$) $\leftarrow$ check if velocity pair is admissible using (3.16)
14:     yawrate($i$) $\leftarrow$ found from (3.18)
15:     dist($i$) $\leftarrow$ distance to avoidance region found by (3.19), (3.41) or
    (3.45)
16:     vel($i$) $\leftarrow$ found from (3.21)
17:     Cost($i$) $\leftarrow \alpha \cdot$ yawrate($i$) $+ \beta \cdot$ dist($i$) $+ \gamma \cdot$ vel($i$)
18: Find $i$ s.t. Cost($i$) = **max**(Cost) **and** admissible($i$)
19: **return** disc_$V(i)$
```

Table 5.2: Constant values used in simulation for algorithms A, B and C. Note that α, β, γ, c and κ are unitless.

Constant	**Value**	**Description**
K_{p_u}	1 kg/s	Surge controller gain
K_{p_r}	2 kg/rad s	Yaw controller gain
Δ	200 m	Lookahead distance
$r_{\mathcal{T}}$	5 m	Antitarget region radius
r_{Ω}	10 m	Avoidance region radius
T_s	1 s	Dynamic Window period
T_a	0.8 s	Time limit for changing rudder angle when calculating the dynamic window
u'_d	9.18 m/s	Desired surge speed for the DW algorithms
t_{pt}	12 s	Time frame of the DW predicted trajectories
α	1	Weight of yaw rate function in the DW algorithms
β	5	Weight of distance function in the DW algorithms
γ	3	Weight of velocity function in the DW algorithms
c	2	Tuning parameter for distance function B
κ	0.5	Tuning parameter for distance function C

The stationary surge speed can be expressed as:

$$\frac{1}{m}F_{X,max} + vr + \frac{1}{m}(X_u u + X_{|u|u}|u|\,u) = 0. \tag{5.11}$$

Lastly, the stationary sway speed can be expressed as:

$$(Y_v + Y_{|v|v}|v|)v + K_{\delta_\psi} u^2 \delta_{\psi,max} - ur = 0, \tag{5.12}$$

where the parameters are listed in Table 5.2.

First, the max surge speed is found by setting $r = v = 0$ in (5.11) and solving for u. Now, r is increased to a small yaw rate and the succeeding rudder angle, surge, and sway speed are found while still applying maximum propeller throttle $F_{X,max}$. The yaw rate is step wise increased until reaching $\pm\delta_{\psi,max}$, which is at the point where the circular top of the cone shape in Figure 5.3 ends. The bottom part of the cone shape is approximated discretely by using numerous values of u in (5.10).

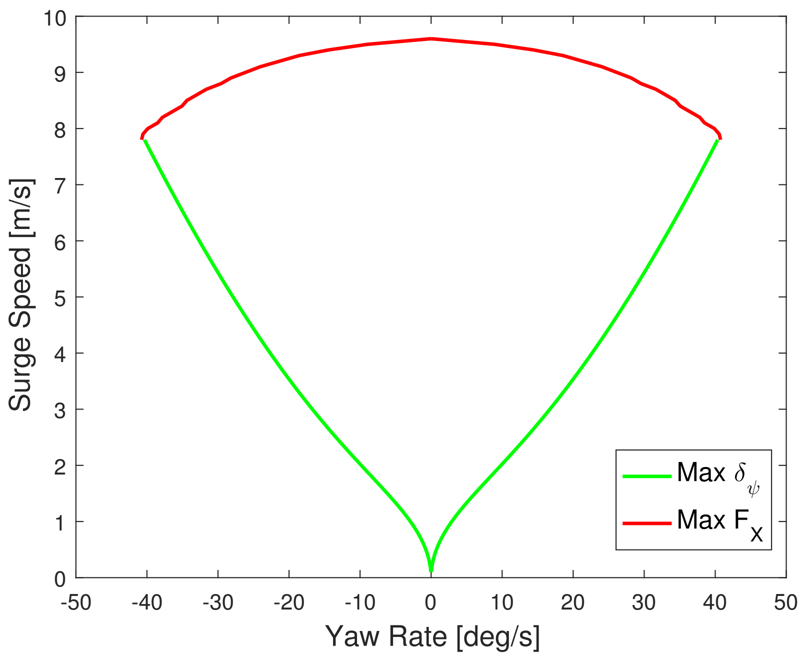


Figure 5.3: Numerical approximated set of possible velocities V_s. The green and red line denotes stationary value when applying maximum rudder angle δ_ψ and maximum propeller throttle, respectively.

Path Tracking

The guidance system tracks a set of way points as described in Section 2.2.1, where the LOS strategy is used to find the desired heading ψ_d. The lookahead distance Δ is a constant listed in Table 5.2. Only two waypoints are used for the simulations in Section 6.1.2. Start and goal represents the waypoints (x_k, y_k) and (x_{k+1}, y_{k+1}) in Section 2.2.2, respectively.

5.2.2 RRT Algorithm

The RRT algorithm is implemented considering the dynamics of the ASV model described in Section 5.1. The tree is expanded by choosing a random state $\mathbf{x}_{rand}$. For simplifying the connection of two states, only the ASV position is considered in the random state $\mathbf{x}_{rand}$, hence the state returned by *RANDOM_STATE*(x_{goal}, ϵ) is expressed as:

$$\mathbf{x}_{rand} = [x_{rand}, y_{rand}]^\top, \tag{5.13}$$

inspired by the implementation of RRT for ASVs in [28]. The parameter $\epsilon \in [0, 1]$ denotes the bias towards making $\mathbf{x}_{rand} = \mathbf{x}_{goal}$. The closest state $\mathbf{x}_{near}$ is then chosen based on translational and rotational distance. The function *NEAREST_NEIGHBOUR*$(\mathbf{x}_{rand}, \boldsymbol{\Upsilon})$ returns the existing state with minimum value of the distance $D(\mathbf{x}_i, \mathbf{x}_{rand})$ expressed as:

$$D(\mathbf{x}_i, \mathbf{x}_{rand}) = \|\mathbf{x}_i - \mathbf{x}_r and\| \cdot \left(1 + \frac{\left|\Theta(\mathbf{x}_i, \mathbf{x}_{rand})\right|}{\pi}\right), \tag{5.14}$$

where $\mathbf{x}_i \in \boldsymbol{\Upsilon}$ is an existing state in the tree, and $\Theta(\mathbf{x}_i, \mathbf{x}_{rand}) \in (-\pi, \pi]$ is the rotational distance from the state $\mathbf{x}_i$ to $\mathbf{x}_{rand}$.

When a the random state $\mathbf{x}_{rand}$ and the nearest neighbour $\mathbf{x}_{near}$ are found, the desired yaw rate r_d is chosen as

$$r_d = k_{r_d} \frac{\Theta(\mathbf{x}_{near}, \mathbf{x}_{rand})}{\pi} \cdot r_{max}(u), \tag{5.15}$$

where $r_{max}(u) > 0$ is the maximum possible ASV yaw rate at surge speed u, and $k_{r_d} > 0$ is a tuning parameter. The desired surge speed u_d is set to a constant value listed in Table 5.3. Based on the desired velocity pair (u_d, r_d), a trajectory is generated towards $\mathbf{x}_{rand}$ by using the trajectory prediction functions (3.37) and (3.38). If the trajectory is collision free and reaches $\mathbf{x}_{rand}$ within a timestep Δt, a new state $\mathbf{x}_{new}$ is generated by $NEW\_STATE(\mathbf{x}_{near}, \mathbf{x}_{rand}, \Delta t, traj)$ and added to the tree $\boldsymbol{\Upsilon}$. If trajectory does not reach $\mathbf{x}_{rand}$ within Δt, the state reached at time $t + \Delta t$ is added to the tree if it is collision free. Finally, *REACHED_GOAL*$(\mathbf{x}_{goal}, \mathbf{x}_{new}, \Delta_{goal})$ is called to check if x_{near} is close enough to the goal to terminate the function. If the state is close enough, the loop ends and a trajectory $\bar{\boldsymbol{\Upsilon}}$ leading to goal is returned. A complete algorithm overview

Algorithm 3: Generating RRT for ASVs

```
1:  $\boldsymbol{\Upsilon} \leftarrow INIT(\mathbf{x}_{init})$
2:  for $k = 1$ to $K$ do
3:      $\mathbf{x}_{rand} \leftarrow RANDOM\_STATE(\mathbf{x}_{goal}, \epsilon)$
4:      $\mathbf{x}_{near} \leftarrow NEAREST\_NEIGHBOUR(\mathbf{x}_{rand}, \boldsymbol{\Upsilon})$
5:      $r_d \leftarrow DESIRED\_ROTATION(\mathbf{X}_{near}, \mathbf{x}_{rand})$
6:      $traj \leftarrow GENERATE\_TRAJECTORY(\mathbf{x}_{rand}, \mathbf{x}_{near}, \Delta t, r_d, u_d)$
7:      $\mathbf{x}_{new} \leftarrow NEW\_STATE(\mathbf{x}_{near}, \mathbf{x}_{rand}, \Delta t, traj)$
8:      if $COLLISION\_FREE(\mathbf{x}_{near}, \mathbf{x}_{new}, \Delta t, traj)$ then
9:          $\boldsymbol{\Upsilon} \leftarrow ADD\_VERTEX(\mathbf{x}_{new})$
10:         $\boldsymbol{\Upsilon} \leftarrow ADD\_EDGE(traj)$
11:         if $REACHED\_GOAL(\mathbf{x}_{goal}, \mathbf{x}_{new}, \Delta_{goal})$ then
12:             $\bar{\boldsymbol{\Upsilon}} \leftarrow COMPLETE\_TRAJECTORY(\boldsymbol{\Upsilon})$
13:         end if
14:     end if
15: end for
16: return $[\boldsymbol{\Upsilon}, \bar{\boldsymbol{\Upsilon}}]$
```

is given in Algorithm 3.

The bias ϵ towards the goal implemented in $RANDOM\_STATE(\mathbf{x}_{goal}, \epsilon)$ greatly improves the expected runtime of the RRT algorithm. However, setting ϵ too high will decrease the RRTs ability to discover unexplored areas, and will hence make it harder to find a way through narrow areas. Figure 5.4 presents an example of a biased RRT.

The RRT algorithm does not consider the limited rudder angle rate, hence the generated trajectories do not guarantee feasibility.

5.2.3 Hybrid COLAV

The hybrid COLAV method is implemented based on the HDW algorithm presented in Chapter 4. The planned path is generated by the deliberate COLAV algorithm RRT. The HDW algorithm is implemented and tested with the obstacle distance functions from both Algorithm B and Algorithm C. The HDW algorithm is implemented with trajectory predictions of two and two velocity pair.

The RRT algorithm generates a new trajectory at the start of every simulation based on the initial ASV state. The trajectory is sent to the HDW algorithm, which uses the planned trajectory as guidance. Then a discrete search space disc_V$_1$ of velocity pair is generated based on current ASV state and surrounding obstacles. A predicted trajectory traj_1 is generated by using (3.38) for every velocity pair in disc_V$_1$. The final state in every predicted trajectory is

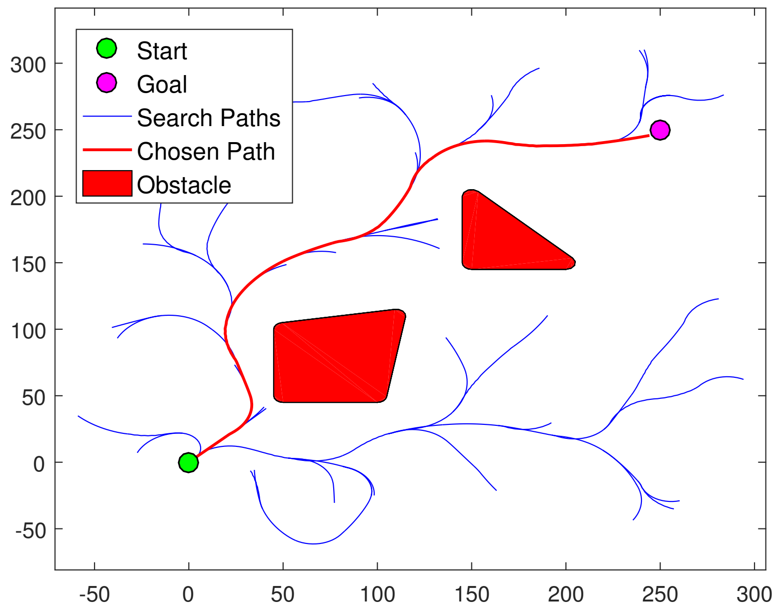


Figure 5.4: Successful RRT with a bias $\epsilon = 0.1$ towards goal.

now used to generate the second discrete search space disc_V$_2$.

For every velocity pair in disc_V$_2$, a new predicted trajectory traj_2 is generated. The predicted trajectories traj_1 and traj_2 are merged into a single trajectory, which is compared with the other predicted trajectories in the cost function (4.10). Finally, the best trajectory is chosen based on the distance function and align function, and the corresponding velocity pair is returned as desired velocity. It is possible to rerun the RRT algorithm during the simulation based on the current ASV state. The HDW algorithm is summarized in Algorithm 4.

Algorithm 4: HDW iteration

```
1:  $\bar{\boldsymbol{\Upsilon}} \leftarrow$ *UPDATE_RRT*$(\boldsymbol{\eta}, \boldsymbol{\nu})$
2:  disc_V$_1$ $\leftarrow$*GENERATE_SEARCHSPACE*$(\boldsymbol{\eta}, \boldsymbol{\nu}, m_1, n_1)$
3:  **for** $i = 1$ to $n \cdot m$ **do**
4:      $[u_1, r_1] \leftarrow$ disc_V$(i)$
5:      traj_1 $\leftarrow$ predicted trajectory for $[u_1, r_1]$ using (3.38)
6:      disc_V$_2$ $\leftarrow$*GENERATE_SEARCHSPACE*$(\boldsymbol{\eta}_p, \boldsymbol{\nu}_p, m_2, n_2)$
7:      **for** $j = 1$ to $m_2 \cdot n_2$ **do**
8:          $[u_2, r_2] \leftarrow$ disc_V$_2(j)$
9:          traj_2 $\leftarrow$ predicted trajectory for $[u_2, r_2]$ using (3.38)
10:         Trajectory$(i, j) \leftarrow$ *MERGE*(traj_1,traj_2)
11:         disc_V$(i, j) \leftarrow$ *MERGE*(disc_V$_1(i)$,disc_V$_2(j)$)
12:         admissible$(i) \leftarrow$ check if velocity pair is admissible using (3.16)
13:         *dist* $\leftarrow$ found using (3.41) or (3.45)
14:         *align* $\leftarrow$ found using (4.7)
15:         Cost$(i, j) \leftarrow \bar{\alpha} \cdot dist - (1 - \bar{\alpha}) \cdot align$, (4.10)
16:     **end for**
17:     **find** $(i, j)$ s.t. Cost$(i, j)$ = max(Cost) **and** admissible$(i, j)$
18: **end for**
19: **return** dist_V$(i, j)$
```

Table 5.3: Constant values used in simulation for the hybrid COLAV method. Note that $\bar{\alpha}$, c, κ and ϵ are unitless. The constants in the feedback linearizing controller of the system are equal to the constants listed in Table 5.2.

Constant	**Value**	**Description**
Δ	200 m	Lookahead distance
$r_{\mathcal{T}}$	5 m	Antitarget region radius
r_{Ω}	10 m	Avoidance region radius
T_s	1 s	Dynamic Window period
T_a	0.8 s	Time limit for changing rudder angle when calculating the dynamic window
u'_d	8.7 m/s	Desired surge speed for the HDW algorithm
t_{pt}	12 s	Time frame of the DW predicted trajectories
$\bar{\alpha}$	0.98	Weight between the HDW align and distance function
c	2	Tuning parameter for distance function B
κ	0.5	Tuning parameter for distance function C
ϵ	0.15	RRT bias towards goal

Chapter 6

Simulation Scenarios and Results

The simulation results are divided into two parts. First, the reactive DW algorithms A, B and C are compared to each other in Section 6.1. Then in Section 6.2, Algorithm B and C are adapted to fit the HDW algorithm described in Chapter 4, and compared to each other.

6.1 DW Algorithms

The algorithms presented in Chapter 3 are simulated in the scenarios described in Section 6.1.1. Several performance metrics are used to evaluate the performance metrics at the end of the section.

6.1.1 Scenarios and Performance Metrics

The scenarios used in the simulations are designed to test the performance of the algorithms for both general and special cases. All the scenarios consist of a path from start to goal that the ASV use for LOS guidance. In addition, the scenarios may have both moving obstacles, which has constant speed and heading, and static obstacles shaped as polygons. If the ASV enters the antitarget region of an obstacle, the simulation assumes it to be a collision. Although most scenarios are fictionally designed to challenge the algorithms, Scenario 7 is inspired by an actual trafficked area to show that the challenging simulation scenarios can be realistic. The scenario is based on the narrow ferry passage around the island Bleikja in Hardangerfjorden. The different scenarios are described in Table 6.1.

The size of the avoidance region $r_{\boldsymbol{\Omega}}$ and antitarget region $r_{\mathcal{T}}$ used in the scenarios are listed in Table 5.2, where $r_{\mathcal{T}}$ is equal to the estimated radius of the

Table 6.1: Overview of simulation scenarios.

Scenario	Type	Description
1	Semi-challenging	To test multitasking with path tracking and collision avoidance, the initial heading leads away from the LOS path. The ASV will now have a significantly large yaw rate when approaching the obstacles.
2	Narrow paths	The start position is inside a narrow environment, which the ASV must maneuver through to reach the goal. This challenges the accuracy of the trajectory prediction and will force the ASV to come close to or enter the avoidance region.
3	Unexpected obstacle	To test the algorithms for unexpected obstacles, the ASV starts heading directly towards a close object.
4	Moving obstacle	To test the ability to avoid moving obstacles, a moving obstacle that interferes with the path is added to a scenario comparable to Scenario 3.
5	Avoidance trap	A trap is used to test if the avoidance region may lead to bad decisions in some scenarios. In addition, a narrow trap is introduced.
6	Distance blinding	The ASV is initiated inside the avoidance region with an oncoming obstacle. This challenges the algorithms ability to exit the avoidance region or avoid moving obstacles while being inside the avoidance region.
7	Narrow passage	A narrow passage with an oncoming obstacle is introduced, inspired by a ferry passage around the island Bleikja in Hardangerfjorden. This demonstrates how the challenges in the scenarios can be found in realistic situations.

vessel footprint and $r_{\boldsymbol{\Omega}}$ is chosen as $r_{\boldsymbol{\Omega}} = 10$ m.

To be able to compare the performance of the algorithms, performance metrics are introduced based on energy consumption, change of control input, and distance from obstacles along the travelled path.

A control input $\bar{\tau}(t)$ is computed as:

$$\bar{\tau}(t) = \sqrt{X(t)^2 + N(t)^2}, \tag{6.1}$$

where $X(t)$ and $N(t)$ is the propeller and rudder throttle, respectively. Given this signal, the integral of absolute differentiated control (IADC), previously used in [14], can be defined as:

$$IADC(t) = \int_0^t \left|\dot{\bar{\tau}}(\sigma)\right| d\sigma, \tag{6.2}$$

which penalizes the actuator wear and tear.

The energy consumption of the ASV is used as a performance metric expressed as:

$$W(t) = \int_0^t P(\sigma) d\sigma, \tag{6.3}$$

where T is the total runtime and $P(\sigma)$ is the force applied by the engine and rudder expressed as:

$$P(t) = \boldsymbol{\tau}(t) \cdot \boldsymbol{\nu}(t). \tag{6.4}$$

The integral of the distance inside the avoidance region (IDI) is expressed as:

$$\text{IDI}(t) = \int_0^T \bar{\lambda}(\sigma) d\sigma, \tag{6.5}$$

where $\bar{\lambda}(\sigma)$ is defined as:

$$\bar{\lambda}(t) = \begin{cases} 1 - \frac{D_{\boldsymbol{\mathcal{T}}}(t)}{r_{\boldsymbol{\Omega}} - r_{\boldsymbol{\mathcal{T}}}}, & \text{if } D_{\boldsymbol{\mathcal{T}}} < r_{\boldsymbol{\Omega}} - r_{\boldsymbol{\mathcal{T}}} \\ 0, & \text{otherwise} \end{cases}, \tag{6.6}$$

which gives the distance inside the avoidance region $\boldsymbol{\Omega}$ if $D_{\boldsymbol{\mathcal{T}}} < r_{\boldsymbol{\Omega}} - r_{\boldsymbol{\mathcal{T}}}$ is satisfied, where $D_{\boldsymbol{\mathcal{T}}}(t)$ is the distance from the ASV to the closest point in the antitarget region $\boldsymbol{\mathcal{T}}$. The metric penalizes staying inside the avoidance region over time and how far inside the avoidance region the ASV is.

The minimum distance D_{min} to the antitarget region in the time frame $[0, T]$ is expressed as:

$$D_{min} = \min_t D_{\boldsymbol{\mathcal{T}}}(t), t \in [0, T], \tag{6.7}$$

and is used as a performance metric penalizing the shortest distance from colliding through the complete run.

Change of control input IADC, energy consumption E, and distance inside avoidance region IDI are desired to be as low as possible, while the minimum distance to obstacles D_{min} is desired to be high. Considering that the algorithm is local, non-optimal, and used as a COLAV algorithm, the minimum distance and the distance inside avoidance region are weighted the most when considering the algorithm performance. The metrics are used as a guide towards a final evaluation of the algorithm performance.

6.1.2 Results

Scenario 1

All the algorithms in Scenario 1 are able to guide the ASV to the goal while staying clear of the avoidance region. As seen in Figure 6.1, starting with a heading away from the LOS path does not cause any problems for the algorithms. As expected algorithms A and B tend to behave identically when not being close to entering or inside the avoidance region. Algorithm C follows almost the same path as algorithms A and B, but the trajectory plot and measurement metrics in figures 6.1 and 6.2 show that it differs slightly from the other algorithms.

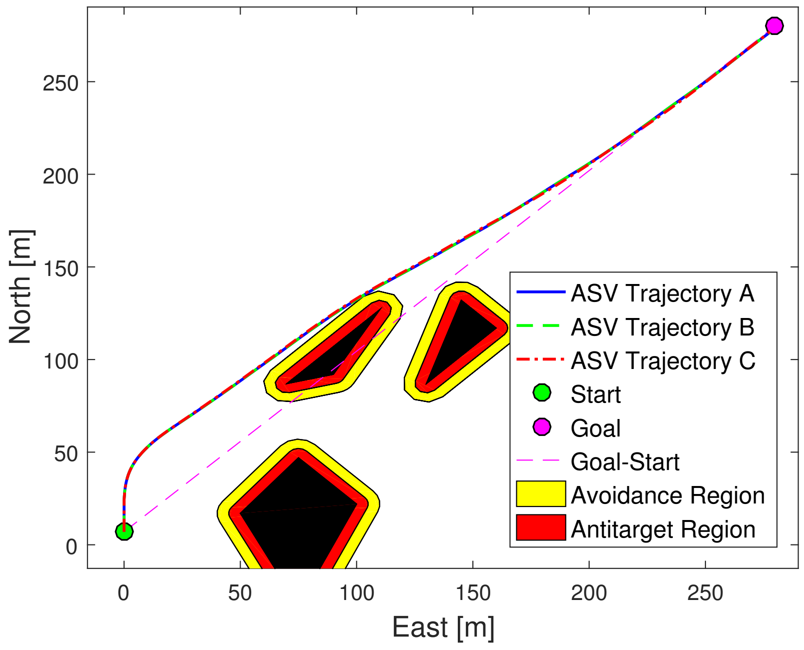
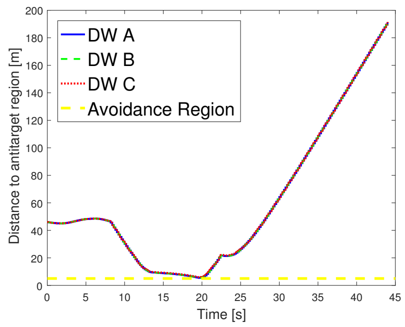
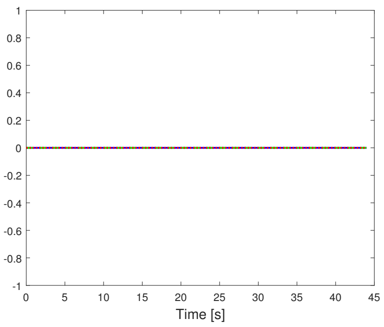


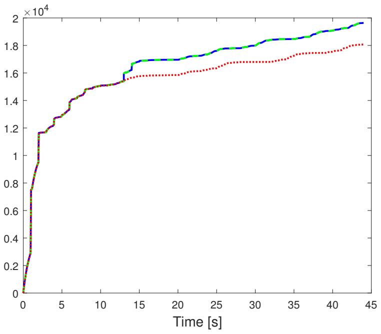
Figure 6.1: Scenario 1 - DW algorithms A, B and C.



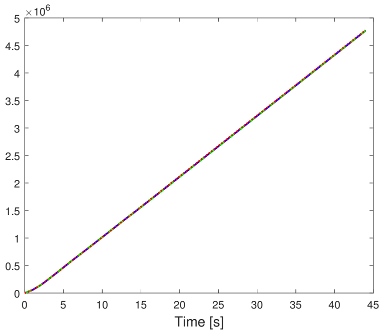
(a) $D_{\mathcal{T}}(t)$



(b) IDI(t)



(c) IADC(t)



(d) W(t)

Figure 6.2: Scenario 1 - Performance metrics. The blue, green and red lines denote the DW algorithms A, B, and C, respectively.

Scenario 2

In this scenario, all the algorithms are forced to travel through a narrow passage. As seen in Figure 6.3, algorithm A is blinded by the avoidance region and chooses the admissible trajectory closest to the desired yaw rate r'_d given by the guidance system. This causes the ASV to travel as close as possible to the antitarget region without colliding but does not collide due to the admissible constraint in the search space. Algorithm B is smoothly travelling through the passage and keeps a relatively good distance to the antitarget. Algorithm C stays almost clear of the avoidance region at the cost of a small overshoot when turning around the obstacle. Algorithm B and C keep approximately equally distances from the obstacles, but Figure 6.4c shows that Algorithm B had a significant higher IADC value. Considering how the distance Algorithm B changes behaviour when being close to the avoidance region, it is reasonable that it causes a greater change in the control inputs.

The minimum distance D_{min} between the ASV and the antitarget region for the different algorithms is 0.11, 3.97 and 3.50 meters for algorithms A, B and C, respectively. The distance of Algorithm A is unacceptable, while algorithms B and C are acceptable considering the narrow passage.

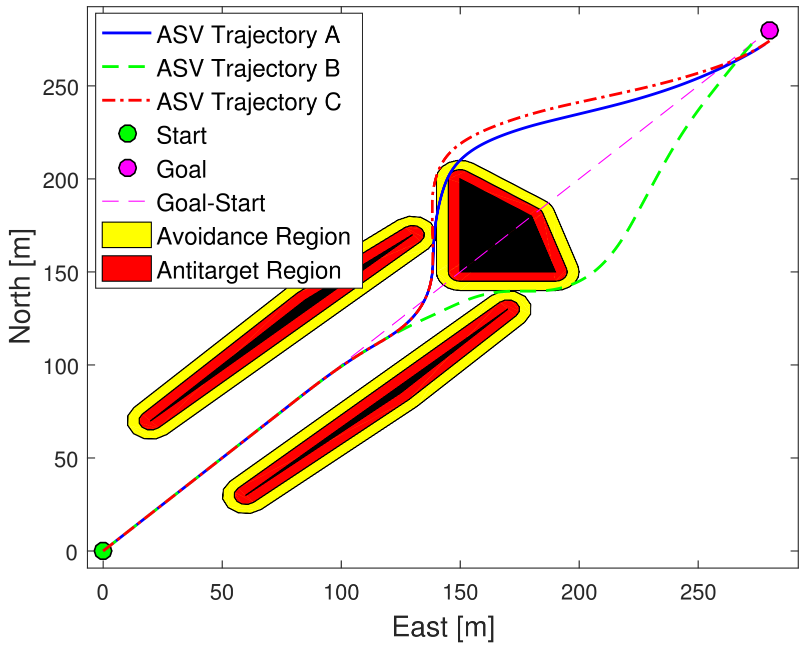
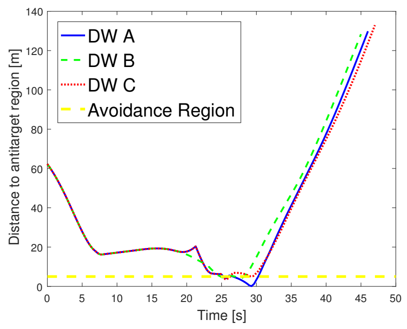
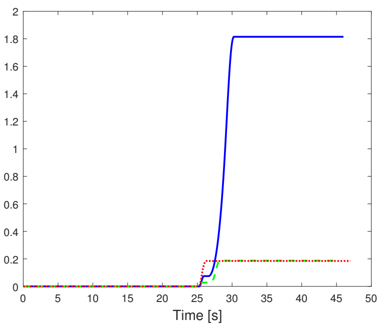


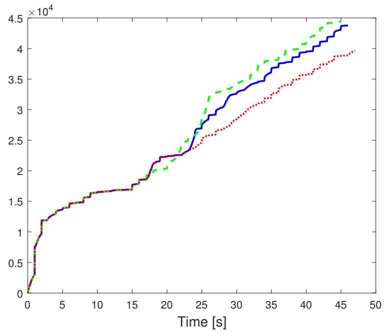
Figure 6.3: Scenario 2 - DW algorithms A, B and C.



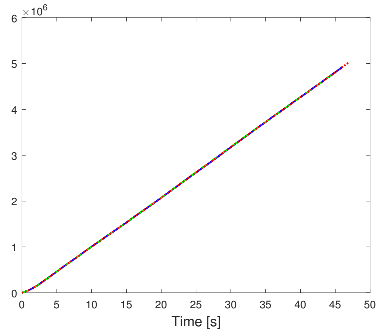
(a) $D_{\mathcal{T}}(t)$



(b) IDI(t)



(c) IADC(t)



(d) W(t)

Figure 6.4: Scenario 2 - Performance metrics. The blue, green and red lines denote the DW algorithms A, B, and C, respectively.

Scenario 3

In Scenario 3, the ASV directly approaches an obstacle at the start but this does not cause difficulties for the algorithms. The algorithms barely cut through the avoidance region when passing around the corner of the obstacles. This is probably caused by the collision check along the predicted trajectory being discrete. The results are presented in Figure 6.5 and the associated performances are presented in Figure 6.6. Algorithms A and B has once again identical trajectories, while Algorithm C has a slightly smoother one.

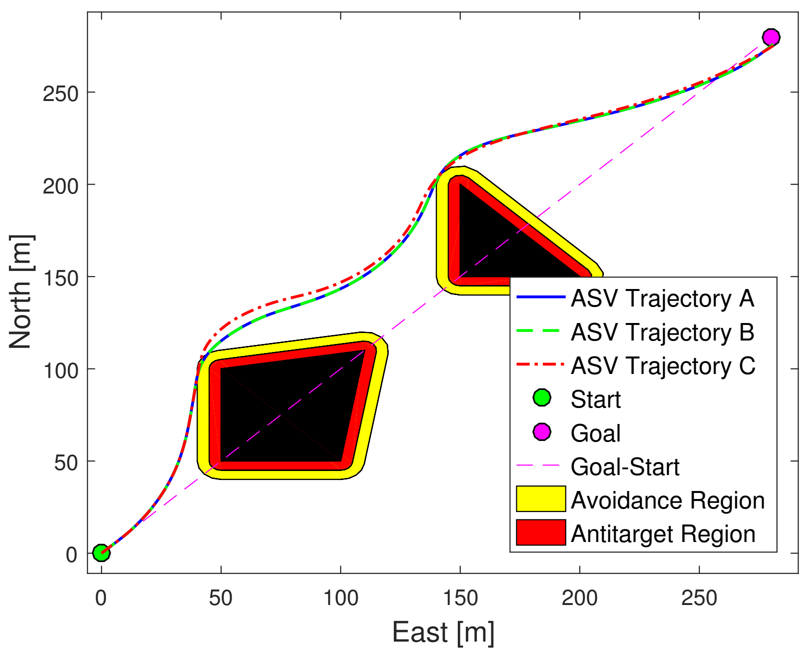
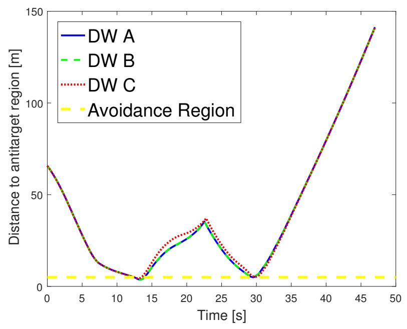
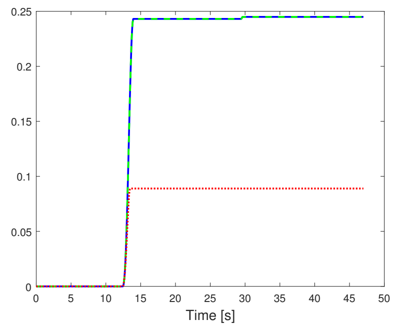


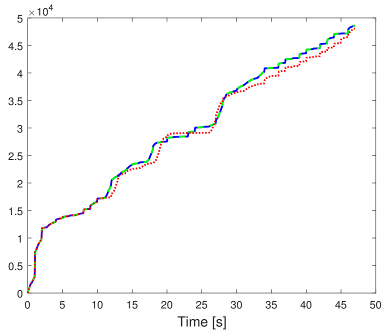
Figure 6.5: Scenario 3 - DW algorithms A, B and C.



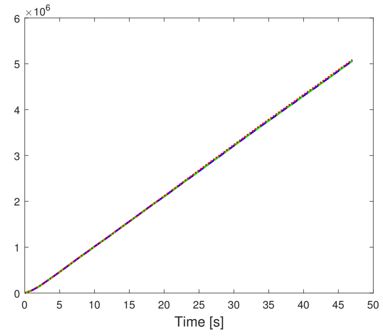
(a) $D_{\mathcal{T}}(t)$



(b) IDI(t)



(c) IADC(t)



(d) W(t)

Figure 6.6: Scenario 3 - Performance metrics. The blue, green and red lines denote the DW algorithms A, B, and C, respectively.

Scenario 4

All the algorithms are successfully able to avoid the moving obstacle in Scenario 4. As seen in Figure 6.7, all algorithms behave identically. At time $t = 23s$ the figure shows how the ASVs barely avoid the avoidance region of the moving obstacle, which demonstrates the precision of the trajectory prediction.

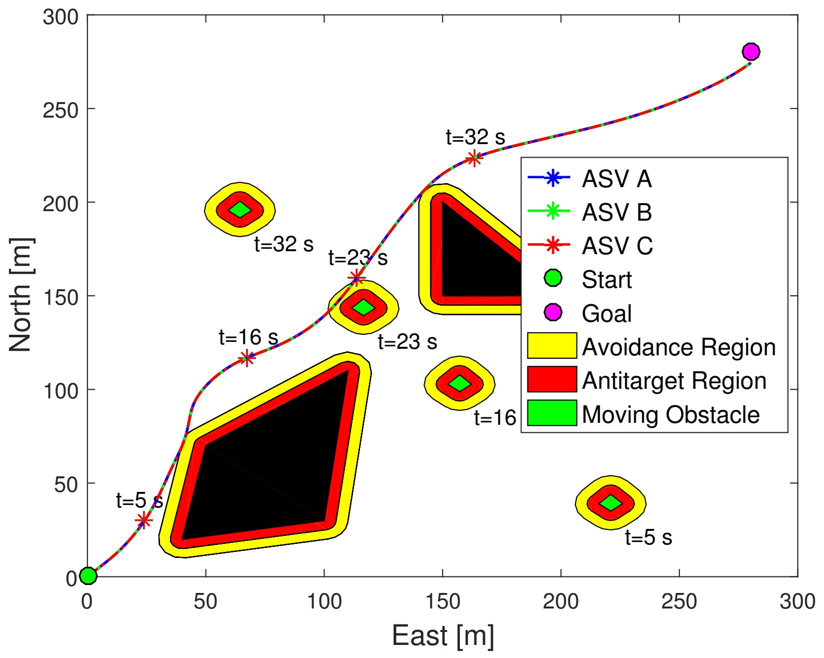
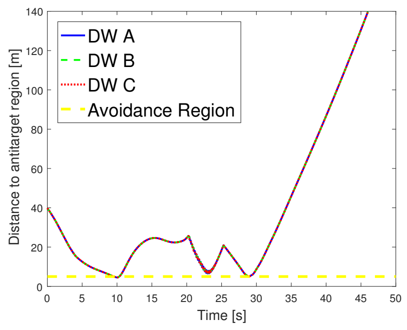
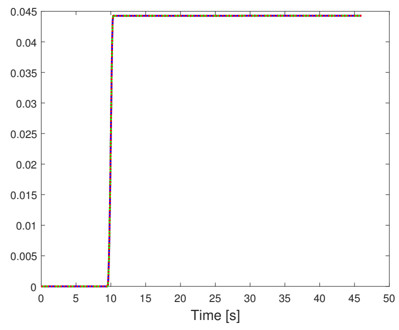


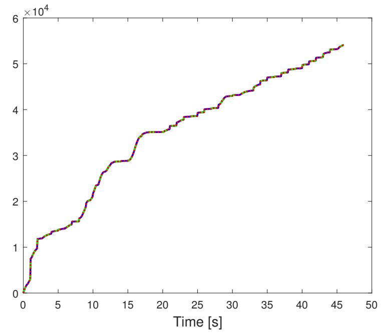
Figure 6.7: Scenario 4 - Snapshots at time 5, 16, 23 and 32.



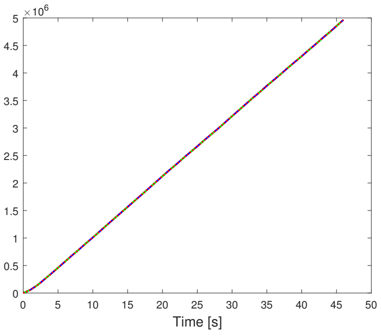
(a) $D_{\boldsymbol{\mathcal{T}}}(t)$



(b) IDI(t)



(c) IADC(t)



(d) W(t)

Figure 6.8: Scenario 4 - Performance metrics. The blue, green and red lines denote the DW algorithms A, B, and C, respectively.

Scenario 5

The advantage of Algorithm C being able to consider the trajectories past the avoidance region is clearly highlighted in this scenario. The ASV is initiated at a path leading directly to the goal. However, the avoidance region is barely touching the path, making algorithms A and B choose any other option in the immediate surroundings where the avoidance region is absent. The path chosen by the algorithms could be a dead end trapping the ASV. Algorithm C weights the portion of the trajectory that resides inside the avoidance region, hence the algorithm does not change direction due to barely sensing the avoidance region. The results are presented in Figure 6.9.

Figure 6.10a presents how close algorithms A and B are to the antitarget region. The minimum distance between the ASV and the antitarget region are for the different algorithms $[D_{min,A}, D_{min,B}, D_{min,C}] = [0.4, 0.11, 5.1]$ and display the dominance of Algorithm C in this scenario.

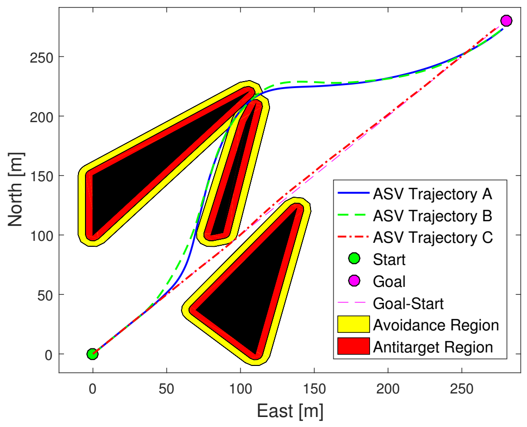
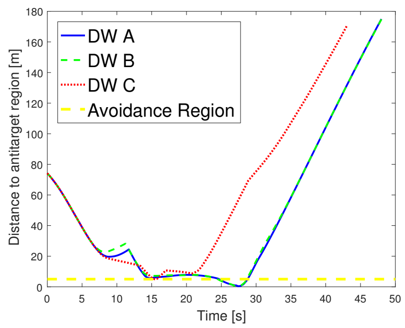
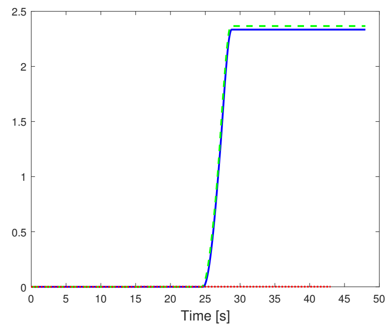


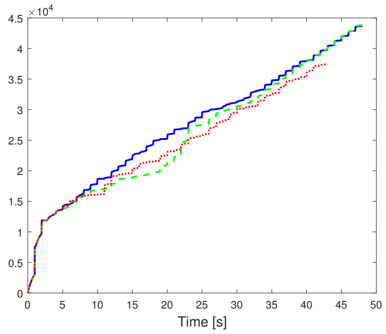
Figure 6.9: Scenario 5 - DW algorithms A, B and C.



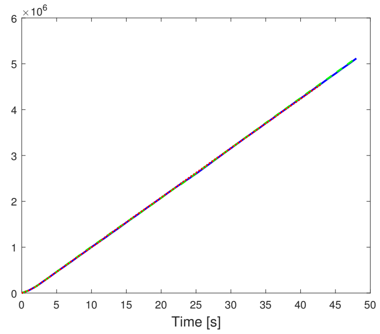
(a) $D_{\boldsymbol{\mathcal{T}}}(t)$



(b) IDI(t)



(c) IADC(t)



(d) W(t)

Figure 6.10: Scenario 5 - Performance metrics. The blue, green and red lines denote the DW algorithms A, B, and C, respectively.

Scenario 6

In Scenario 6, the ASV is initiated inside the avoidance region. As Figure 6.11 shows, algorithms B and C exit the avoidance region easily at the start. Once the ASV is outside the avoidance region, avoiding the approaching obstacle is a simple task for the algorithms. Algorithm A is, however, stuck in the avoidance region and is therefore not able to avoid the approaching obstacle. Algorithms B and C keep the ASV at an approximately equal minimum distance from the moving obstacle, although Algorithm C follows a smoother trajectory as Figure 6.12c illustrates.

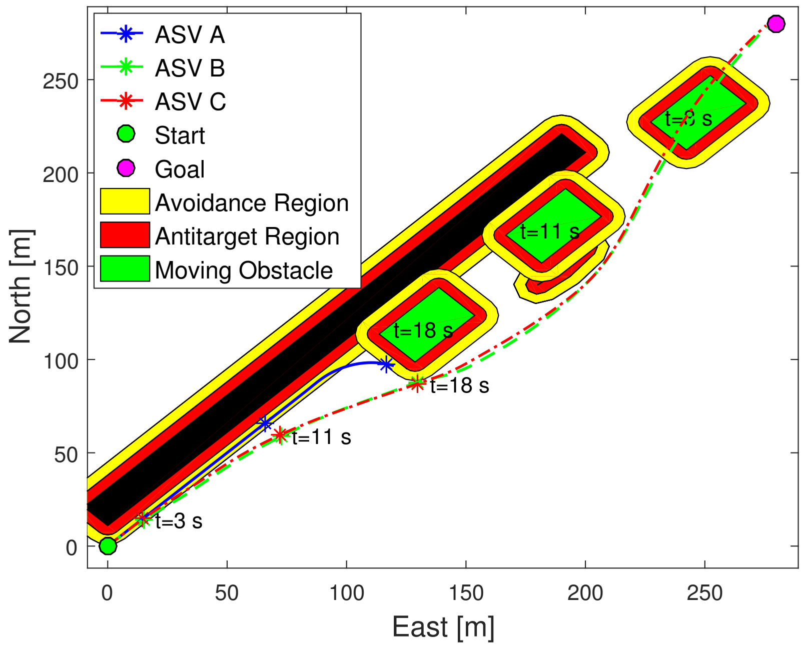
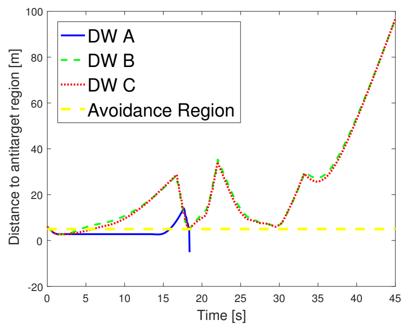
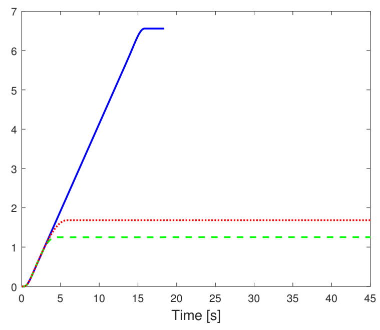


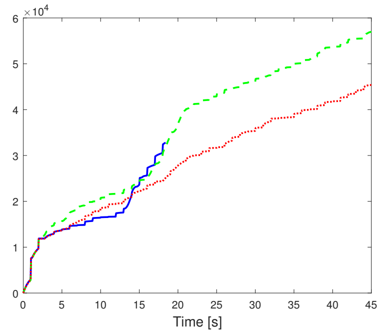
Figure 6.11: Scenario 6 with moving obstacles - Snapshots at time 3, 11 and 18.



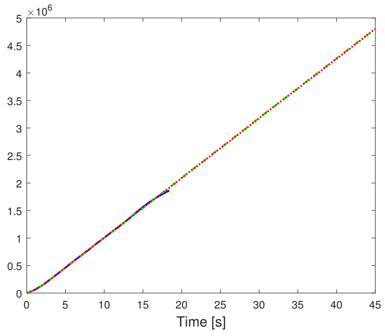
(a) $D_{\mathcal{T}}(t)$



(b) IDI(t)



(c) IADC(t)



(d) W(t)

Figure 6.12: Scenario 6 - Performance metrics. The blue, green and red lines denote the DW algorithms A, B, and C, respectively.

Scenario 7

Scenario 7 is inspired by a narrow ferry passage in Hardangerfjorden. If the ASV does not encounter a moving obstacle (ferry), it has no problem moving through the passage, as seen in Figure 6.13. However, Figure 6.14 demonstrates how a moving obstacle coming in the opposite direction causes problems if the ASV is trying to pass at the same time. At time $t = 11s$, Algorithm A is in the same state as Algorithm B. However, Algorithm A is unable to consider the trajectories past the first entrance of the avoidance region, and is therefore unable to choose the trajectories avoiding the obstacle, as seen at time $t = 17s$. When using Algorithm B, the ASV encounters the moving obstacle in the same state as when using Algorithm A but is able to choose a trajectory leading out of the avoidance region. Algorithm C does not have to be close to the obstacle to consider the trajectories past the avoidance region, and can early choose a path that stays clear of the moving obstacle, at the risk of entering the avoidance region of another obstacle.

The scenario without moving obstacles is an easy challenge for the algorithms, although it should once again be noted how the algorithms behave identically when the ASV keeps some distance to the avoidance region. The minimum distances for the ASV in the scenario with moving obstacles are $[D_{min,A}, D_{min,B}, D_{min,C}] = [0, 0.7, 5.3]$, which shows that Algorithm B leads the ASV dangerously close to the antitarget region.

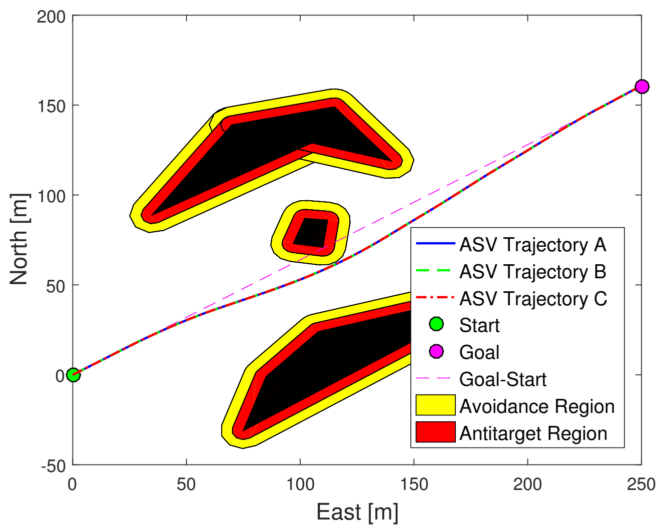


Figure 6.13: Scenario 7 - DW algorithms A, B and C.

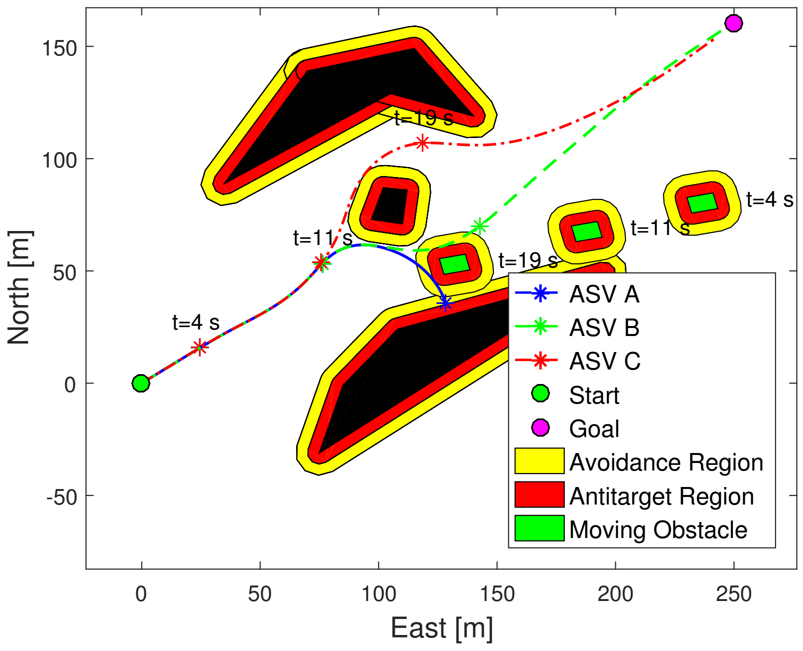
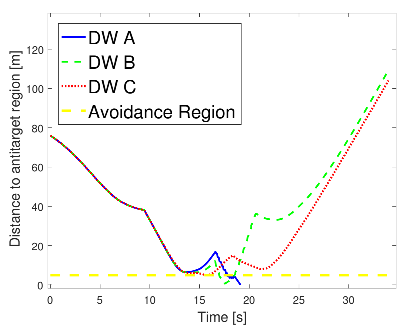
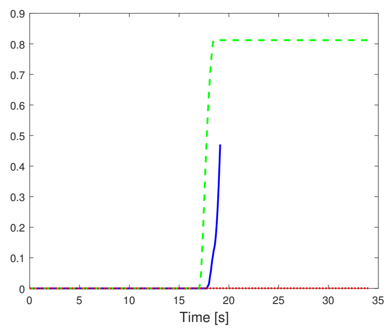


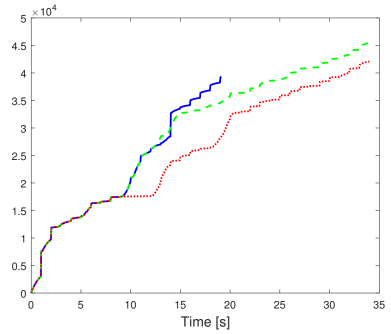
Figure 6.14: Scenario 7 with moving obstacles - Snapshots at time 4, 11 and 19.



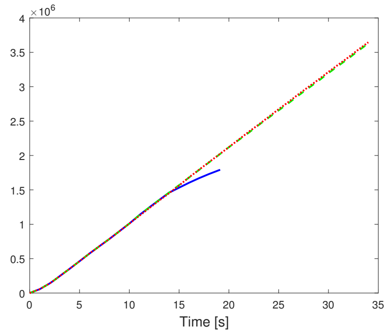
(a) $D_{\boldsymbol{\mathcal{T}}}(t)$



(b) IDI(t)



(c) IADC(t)



(d) W(t)

Figure 6.15: Scenario 7 - Performance metrics. The blue, green and red lines denote the DW algorithms A, B, and C, respectively.

Summary of Results

The simulation results presented in this section are evaluated based on the performance metrics described in Section 6.1.1, and by qualitatively evaluating the ASV trajectories. A complete overview of the performance metrics at the end of each simulation is given in Table 6.2.

Table 6.2: Performance metric values of DW algorithms A, B and C at simulation end time T. The bold values represent the best performance in each metric for every scenario. If the list does not show results for an algorithm, it indicates that the algorithm collided.

Scenario	**Algorithm**	$\boldsymbol{D_{min}}$	**IDI**	**IADC**	W
1	A	5.4	0	$2.0 \cdot 10^4$	$4.8 \cdot 10^6$
	B	5.4	0	$2.0 \cdot 10^4$	$4.8 \cdot 10^6$
	C	**5.8**	0	$\mathbf{1.8 \cdot 10^4}$	$4.8 \cdot 10^6$
2	A	0.1	1.8	$4.4 \cdot 10^4$	$4.9 \cdot 10^6$
	B	**4.0**	**0.2**	$4.4 \cdot 10^4$	$\mathbf{4.8 \cdot 10^6}$
	C	3.5	**0.2**	$\mathbf{4.0 \cdot 10^4}$	$5.0 \cdot 10^6$
3	A	3.6	0.2	$4.9 \cdot 10^4$	$5.0 \cdot 10^6$
	B	3.6	0.2	$4.9 \cdot 10^4$	$5.0 \cdot 10^6$
	C	**4.3**	**0.1**	$\mathbf{4.8 \cdot 10^4}$	$5.0 \cdot 10^6$
4	A	4.5	0.0	$5.4 \cdot 10^4$	$5.0 \cdot 10^6$
	B	4.5	0.0	$5.4 \cdot 10^4$	$5.0 \cdot 10^6$
	C	4.5	0.0	$5.4 \cdot 10^4$	$5.0 \cdot 10^6$
5	A	0.4	2.3	$4.4 \cdot 10^4$	$5.1 \cdot 10^6$
	B	0.1	2.4	$4.4 \cdot 10^4$	$5.1 \cdot 10^6$
	C	**5.1**	**0**	$\mathbf{3.7 \cdot 10^4}$	$\mathbf{4.6 \cdot 10^6}$
6	A	-	-	-	-
	B	2.7	**1.3**	$5.7 \cdot 10^4$	$4.8 \cdot 10^6$
	C	2.7	1.7	$\mathbf{4.5 \cdot 10^4}$	$4.8 \cdot 10^6$
7	A	6.0	0	$3.3 \cdot 10^4$	$3.5 \cdot 10^6$
	B	6.0	0	$3.3 \cdot 10^4$	$3.5 \cdot 10^6$
	C	6.0	0	$3.3 \cdot 10^4$	$3.5 \cdot 10^6$
7 mov	A	-	-	-	-
	B	0.7	0.8	$4.5 \cdot 10^4$	$3.6 \cdot 10^6$
	C	**5.0**	**0**	$\mathbf{4.2 \cdot 10^4}$	$3.6 \cdot 10^6$

Based on the performance metrics and qualitatively evaluation, Table 6.3 is formed to give an overview of the simulation results where every algorithm in every scenario is rated as either collided, bad, good or perfect. A result is marked as collided if the ASV enters the antitarget region $\boldsymbol{\mathcal{T}}$. A path leading to the goal, but barely avoiding a collision is marked as bad, while good and

Table 6.3: Overview of simulation results for DW algorithms. Results are rated as either collided, bad, good or perfect.

Scenario	**Algorithm A**	**Algorithm B**	**Algorithm C**
1	Perfect	Perfect	Perfect
2	Bad	Perfect	Perfect
3	Good	Good	Good
4 mov	Good	Good	Good
5	Bad	Bad	Perfect
6 mov	Collided	Perfect	Perfect
7	Perfect	Perfect	Perfect
7 mov	Collided	Bad	Perfect

perfect denotes increasing performances in the same order.

Considering that Algorithm B always performs equally or better than Algorithm A, Algorithm B is the optimal choice between the two algorithms. This is emphasized by the fact that choosing a high tuning parameter c in (3.40) for Algorithm B causes the behaviour outside the avoidance region to be identical to Algorithm A, but motivates the ASV to exit the avoidance region if entered. Note that Algorithm B consistently returns a higher or equal IADC value than Algorithm C. This is reasonable considering that the cost function in Algorithm B changes behaviour when going close to or inside the avoidance region.

Algorithm C scores significantly better on the metric performance compared to the other algorithms. Furthermore, the algorithm delivers good results consistently, which is a major deal breaker for COLAV algorithms considering the potential cost of a collision. Algorithm C is less hesitant to travel through the avoidance region compared to the other algorithms. Although this may seem like a bad property for a COLAV algorithm, the algorithm is only less hesitant to go through the avoidance region if the predicted trajectory leads out of it again. Hence, if the trajectory going into the avoidance region does not lead out again, Algorithm C is just as cautious about entering the region as the other algorithms. This trait appears valuable if the algorithm is merged with a deliberate COLAV method leading the ASV safely through the avoidance region to avoid collisions.

Based on the simulation results in the scenarios, Algorithm C is the superior among the tested algorithms. Both algorithms B and C will, however, be tested as part of a hybrid COLAV method in Section 6.2.

Algorithms B and C are introduced to cope with the weaknesses of Algorithm A. Clearly, both algorithms B and C are superior to Algorithm A in the tested scenario, especially when operating close to or inside the avoidance region.

6.2 Hybrid COLAV

This section presents simulation results of the hybrid COLAV method with RRT + HDW introduced in Chapter 4. The simulations include a few additional scenarios and performance metrics compared to the reactive DW simulations.

6.2.1 Scenarios and Performance Metrics

In addition to the scenarios used in Section 6.1.1, two new scenarios are introduced for testing the HDW algorithm. A description of the scenarios are presented in Table 6.4.

Table 6.4: Overview of simulation scenarios for the HDW algorithm.

Scenario	**Type**	**Description**
8	Unknown dynamic obstacle	A moving obstacle unknown to the RRT algorithm is introduced. Hence, the reactive algorithm is forced to guide the ASV out of collision, which may lead it far from the planned path.
9	Local minima	Scenario 9 is introduced solely to show the advantage of a deliberate method compared to a reactive method for avoiding local minima.

Considering that the hybrid method consists of a deliberate method generating a planned trajectory and a reactive method guided by it, an error $e(t) \geq 0$ is introduced as:

$$e(t) = \left\| \boldsymbol{x}_d(t) - \boldsymbol{x}(t) \right\|, \tag{6.8}$$

where $\boldsymbol{x}(t)$ denotes the position of the ASV, and $\boldsymbol{x}_d(t)$ denotes the desired position of the ASV at time t. The integral of the absolute error (IAE) used in [35] is then used as a performance metric expressed as:

$$\text{IAE}(t) = \int_0^t \left| e(\sigma) \right| d\sigma. \tag{6.9}$$

The IAE(t) metric displays the absolute value of the error $e(t)$ during the simulations.

6.2.2 Results

The HDW algorithm is tested with the distance function from Algorithm B and Algorithm C. In addition, the HDW algorithm is tested with $\bar{\alpha} = 0$ so that only the trajectory alignment (TA) function is weighted, and will be denoted HDW TA in the simulation results. When $\bar{\alpha} = 0$, the algorithm cares only about following the planned trajectory but will only select velocity pair that lies within the search space. This can be compared to a global algorithm using a trajectory tracker that only chooses velocity pairs that do not directly lead to collisions. The planned trajectory used in the simulations are generated by the RRT algorithm.

The main goal is to test the HDW algorithm ability to work as a trajectory tracker while avoiding collisions.

Scenario 1

Scenario 1 ensures that the ASV has a nonzero yaw rate when approaching the obstacles. The RRT algorithm generates a planned trajectory which turns smoothly around the obstacles and causes the ASV not to approach the obstacles. The algorithms have slightly different behaviours as presented in figures 6.16 and 6.17, but they are all able to follow the planned trajectory.

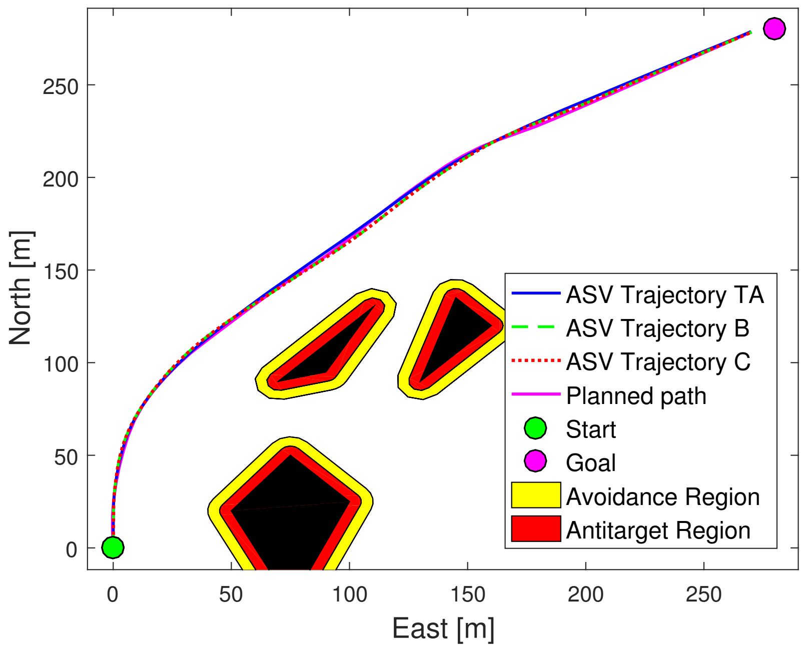
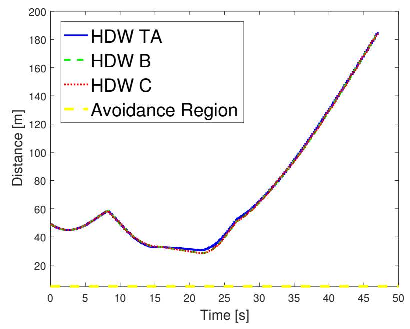
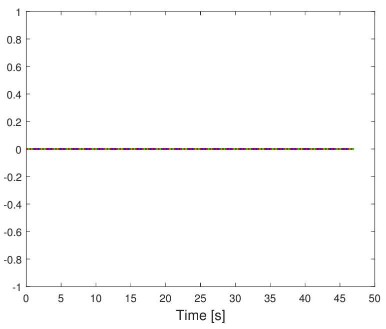


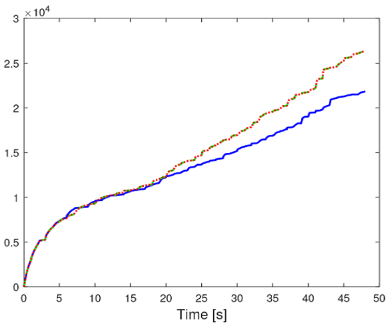
Figure 6.16: Scenario 1 - Hybrid method with RRT + HDW.



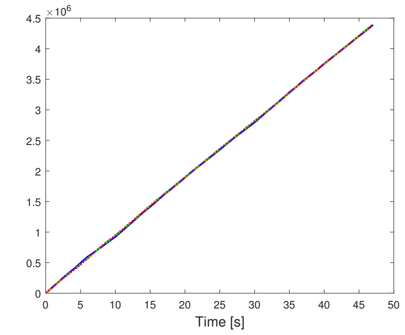
(a) $D_{\mathcal{T}}(t)$



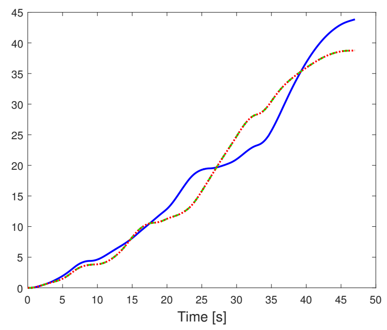
(b) IDI(t)



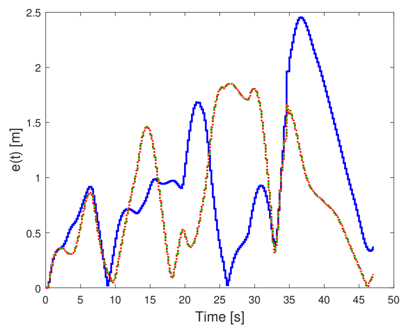
(c) IADC(t)



(d) W(t)



(e) IAE(t)



(f) $e(t)$

Figure 6.17: Scenario 1 - Performance metrics. The blue, green and red lines denote the HDW algorithms TA, B, and C, respectively.

Scenario 2

In Scenario 2, the planned trajectory is turning right away to dodge the narrow path. The prediction seems to overestimate the responsiveness of the ASV yaw rate, which causes the ASV to get an overshoot in the turn. This overshoot leads close to the avoidance region, hence HDW B and HDW C favor dodging the avoidance region over following the planned trajectory as seen in Figure 6.18. HDW TA cares only about following the planned trajectory and cuts through the avoidance region, which causes it to enter the antitarget region and collide.

Although HDW B and HDW C falls far behind from the planned trajectory, Figure 6.19f shows that the vehicle is able to catch up with the trajectory. Finally, HDW B and HDW C avoids the avoidance region and are able to catch up with the planned trajectory, while HDW TA enters the antitarget region.

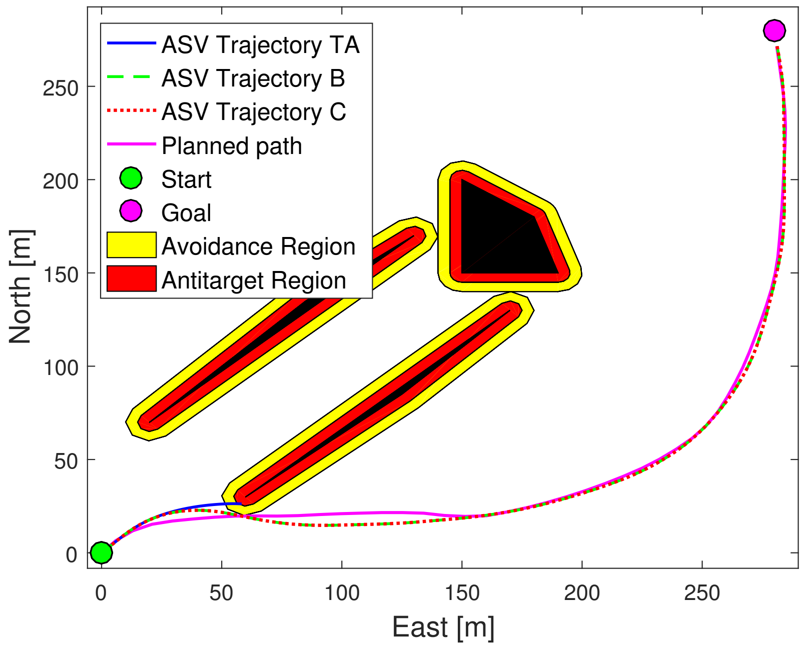
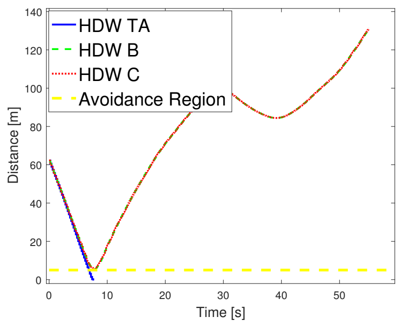
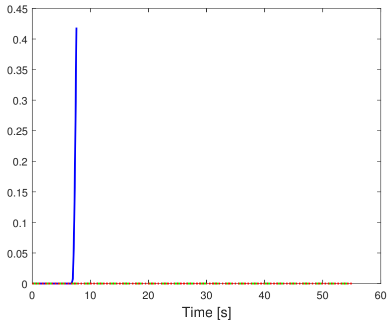


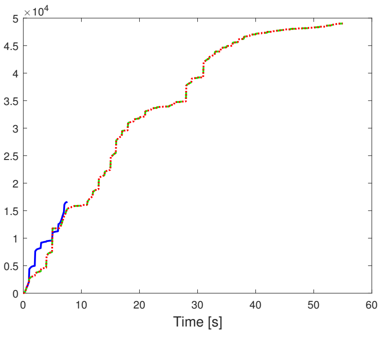
Figure 6.18: Scenario 2 - Hybrid method with RRT + HDW.



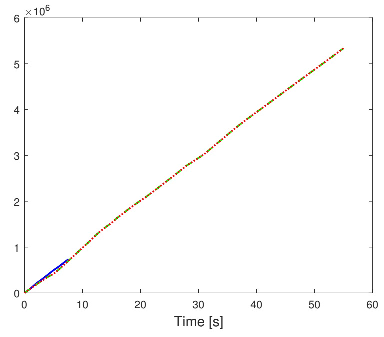
(a) $D_{\mathcal{T}}(t)$



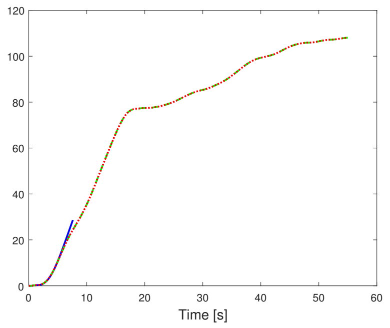
(b) IDI(t)



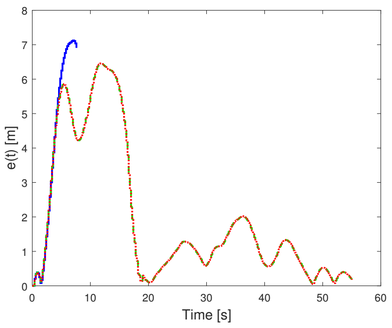
(c) IADC(t)



(d) W(t)



(e) IAE(t)



(f) $e(t)$

Figure 6.19: Scenario 2 - Performance metrics. The blue, green and red lines denote the HDW algorithms TA, B, and C, respectively.

Scenario 3

Scenario 3 challenges the algorithms by making the ASV initiate heading directly towards an obstacle. The planned trajectory turns immediately and overestimates the responsiveness of the ASV yaw rate. Besides from this, all algorithms leads the ASV safe around the obstacles. Figures 6.20 and 6.21 shows that HDW algorithms B and C yield identical results and are able to follow the planned trajectory. HDW TA has a similar trajectory but differs slightly from the other ASVs.

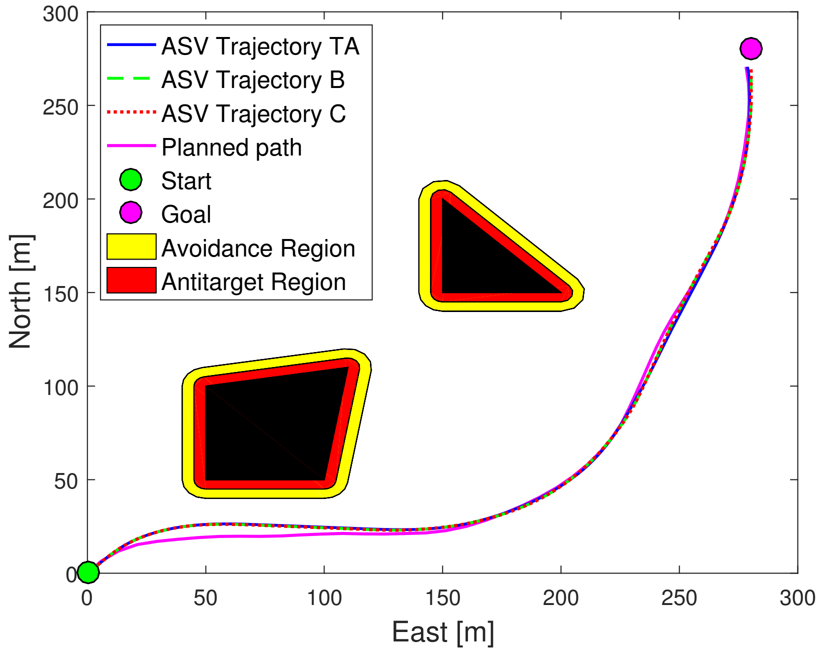
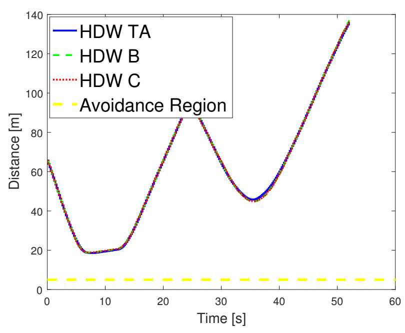
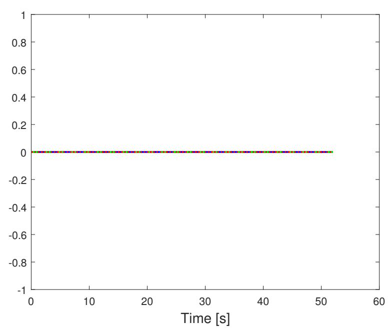


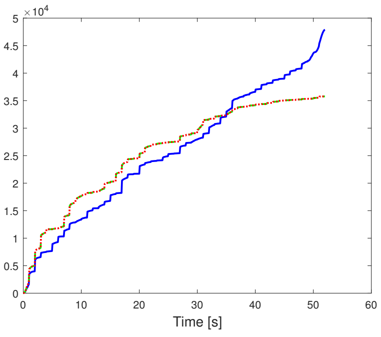
Figure 6.20: Scenario 3 - Hybrid method with RRT + HDW.



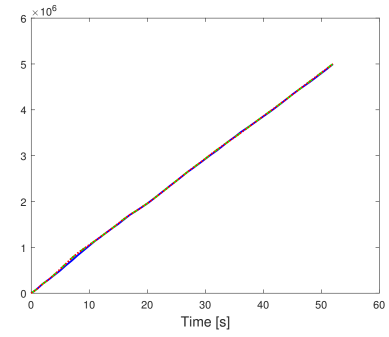
(a) $D_{\mathcal{T}}(t)$



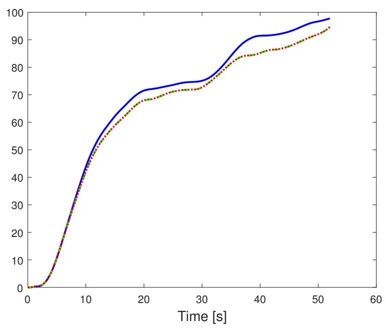
(b) IDI(t)



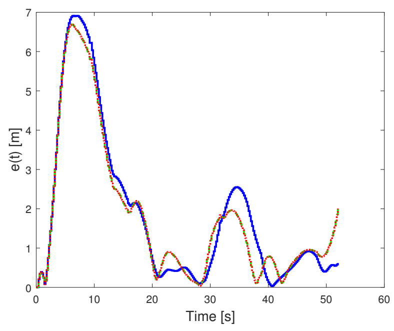
(c) IADC(t)



(d) W(t)



(e) IAE(t)



(f) $e(t)$

Figure 6.21: Scenario 3 - Performance metrics. The blue, green and red lines denote the HDW algorithms TA, B, and C, respectively.

Scenario 4

Scenario 4 introduces a moving obstacle which crosses the path between start and goal. The RRT algorithm is able to generate a trajectory that avoids the obstacle. After the overshoot in the first turn in Figure 6.22, all algorithms are able to follow the planned trajectory. The planned trajectory leads through the avoidance region of both the moving and the static obstacle. HDW TA is led through the avoidance region both places, which Figure 6.23b confirms. HDW B and HDW C barely enter the avoidance region of the moving obstacle.

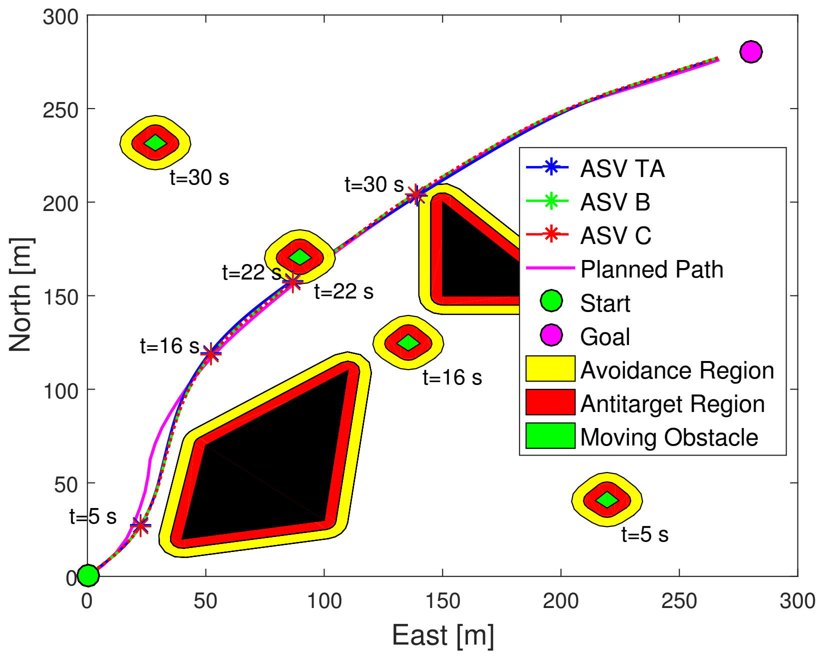
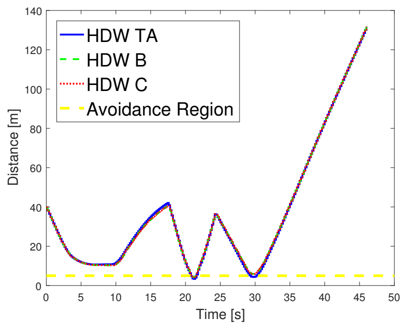
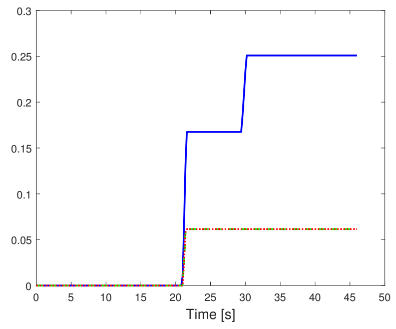


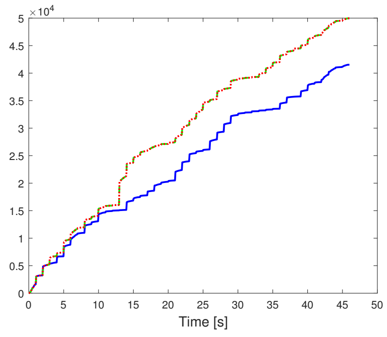
Figure 6.22: Scenario 4 - Hybrid method with RRT + HDW.



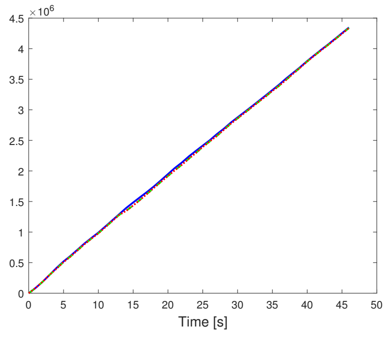
(a) $D_{\mathcal{T}}(t)$



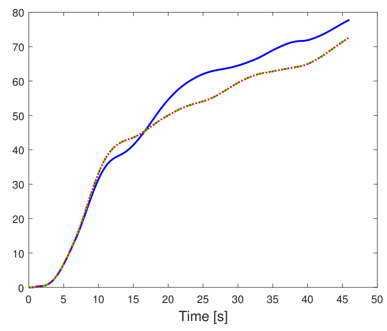
(b) IDI(t)



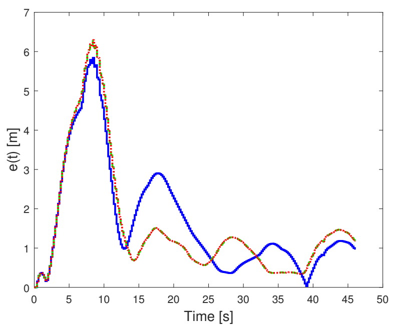
(c) IADC(t)



(d) W(t)



(e) IAE(t)



(f) $e(t)$

Figure 6.23: Scenario 4 - Performance metrics. The blue, green and red lines denote the HDW algorithms TA, B, and C, respectively.

Scenario 5

The reactive algorithms A and B had problems evaluating trajectories past the avoidance region in Scenario 5 in Section 6.1.2. When using the hybrid COLAV method with RRT and HDW, all the algorithms are able to follow the planned trajectory and avoid going through the narrow path. By looking closely on Figure 6.24, we can see that HDW TA slightly crosses the avoidance region, which is confirmed in the metrics in Figure 6.25. HDW B and HDW C avoids touching the avoidance region at the cost of a higher value of IADC. The ASV behaviour varies early in the simulation due to planned trajectory crossing through the avoidance region.

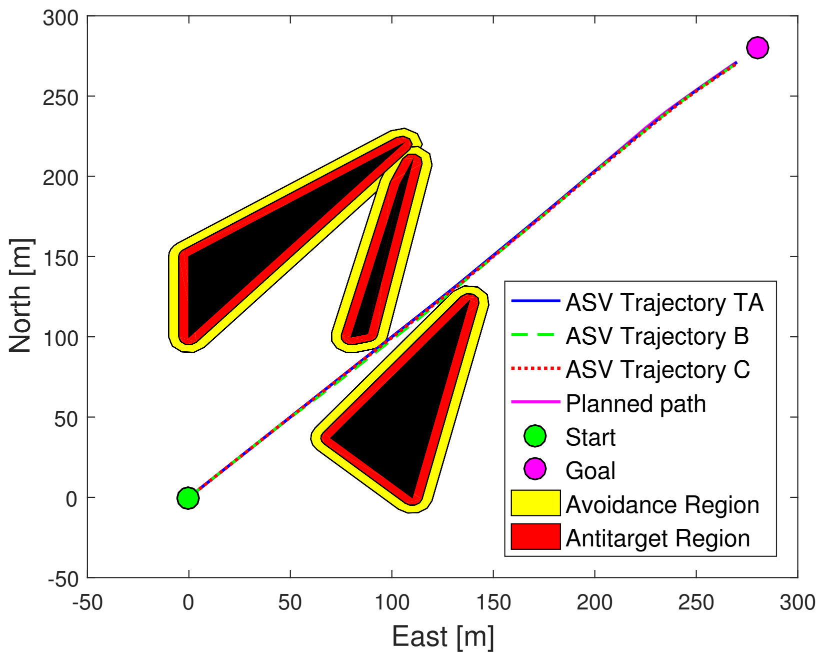


Figure 6.24: Scenario 5 - Hybrid method with RRT + HDW.

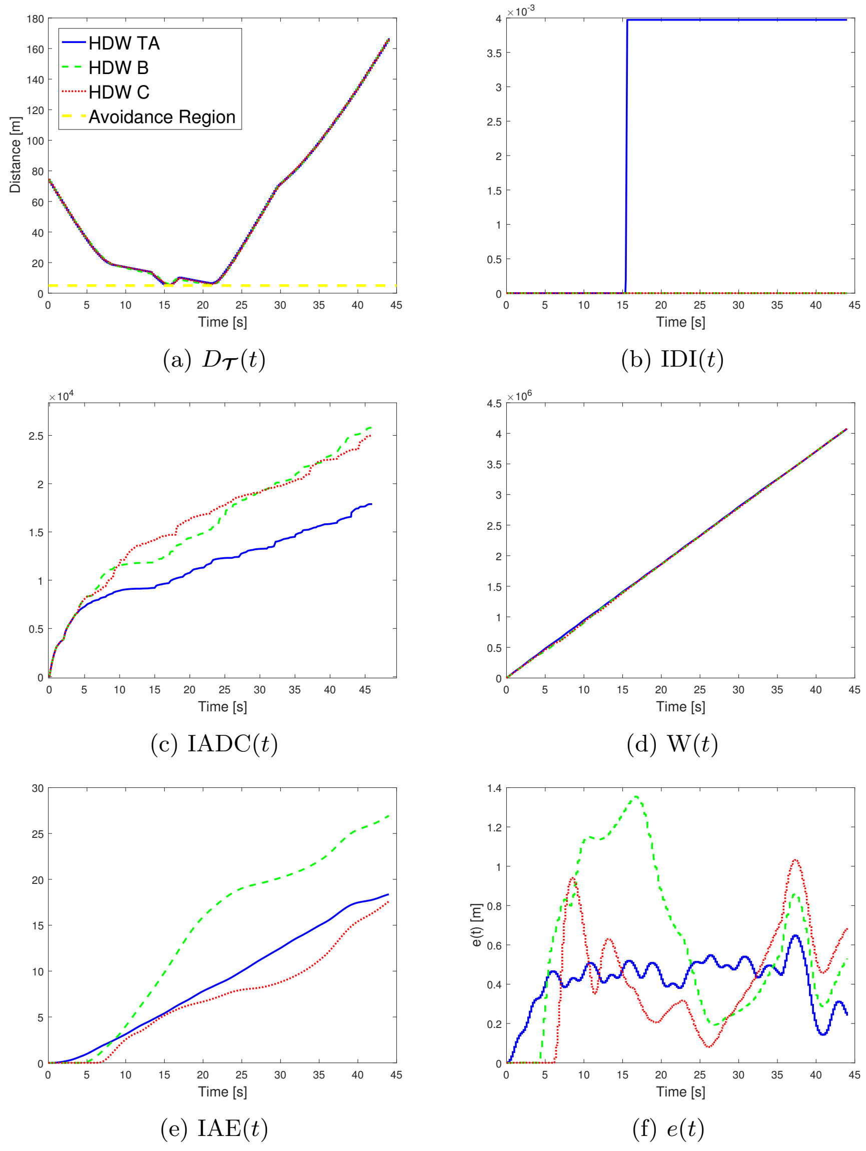


Figure 6.25: Scenario 5 - Performance metrics. The blue, green and red lines denote the HDW algorithms TA, B, and C, respectively.

Scenario 6

In this scenario, the ASV initiates inside the avoidance region with an oncoming moving obstacle. Both HDW B and HDW C is led out of the avoidance region, away from the planned trajectory. Distance function B is more motivated to leave the avoidance region quickly than distance function C, and is therefore moving further out of the avoidance region. This comes at the cost of travelling further away from the planned trajectory, compared to HDW C and HDW TA.

Figure 6.26 presents snapshots at different times in the simulation and shows that HDW TA is close to the antitarget region of the moving obstacle. Figure 6.27b confirms that HDW TA is inside the avoidance region around time $t = 18s$. The HDW function is equally good at avoiding the moving obstacle when using either distance function B or C. Distance function C does, however, follow the planned trajectory more smoothly after exiting the avoidance region in the beginning and is hence the more favorable in this scenario.

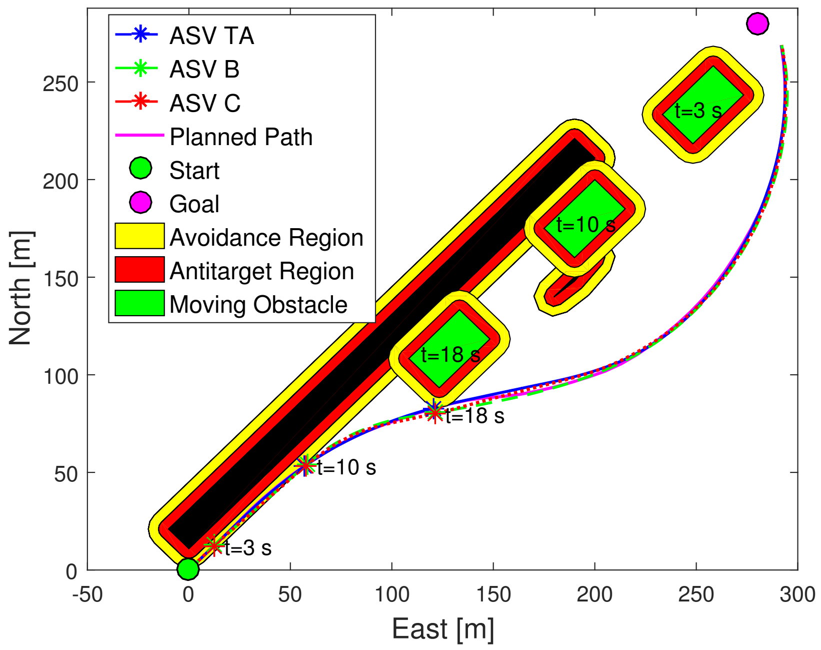
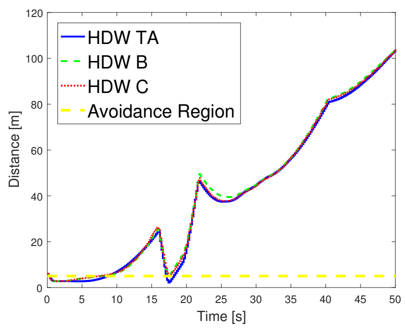
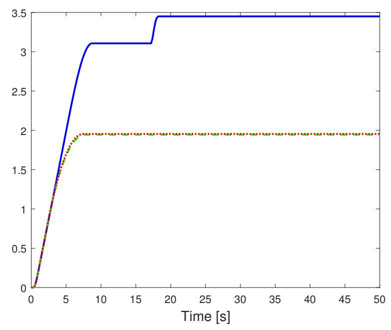


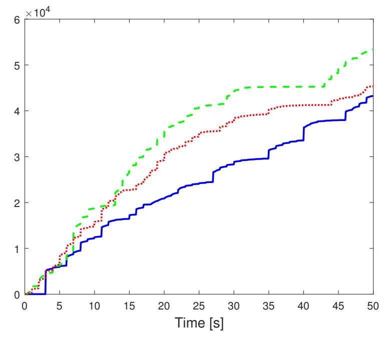
Figure 6.26: Scenario 6 - Hybrid method with RRT + HDW.



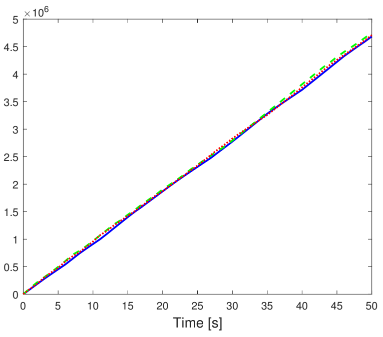
(a) $D_{\mathcal{T}}(t)$



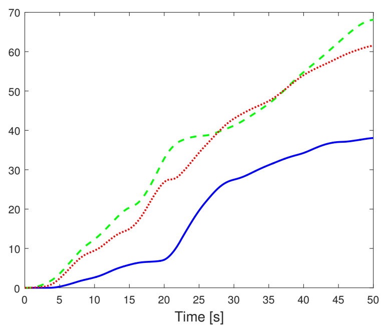
(b) IDI(t)



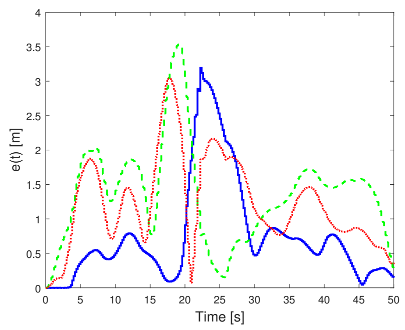
(c) IADC(t)



(d) W(t)



(e) IAE(t)



(f) $e(t)$

Figure 6.27: Scenario 6 - Performance metrics. The blue, green and red lines denote the HDW algorithms TA, B, and C, respectively.

Scenario 7

The RRT algorithm generates a path through Scenario 7 that leads the ASV through the avoidance region, close to both a static and a dynamic obstacle. HDW B leaves the planned trajectory to avoid entering the avoidance region, but once the ASV is close to the avoidance region HDW B can see a path leading trough the region. This leads HDW B closer to the planned trajectory but causes it to almost enter the antitarget region. HDW C and HDW TA follows the trajectory at the cost of touching the avoidance region. Considering that HDW B almost enters the antitarget region, the performance is bad. In addition, HDW B is never able to fully catch up with the planned trajectory. Algorithm C is less hesitant to enter the avoidance region if the path leads through it, which is illustrated in the results in Figure 6.28. Figure 6.29b shows how HDW TA enters deeper into the avoidance region than HDW C.

To sum up, HDW C avoids the avoidance region as much as possible while still being able to follow the planned trajectory, hence it performs better than the other algorithms.

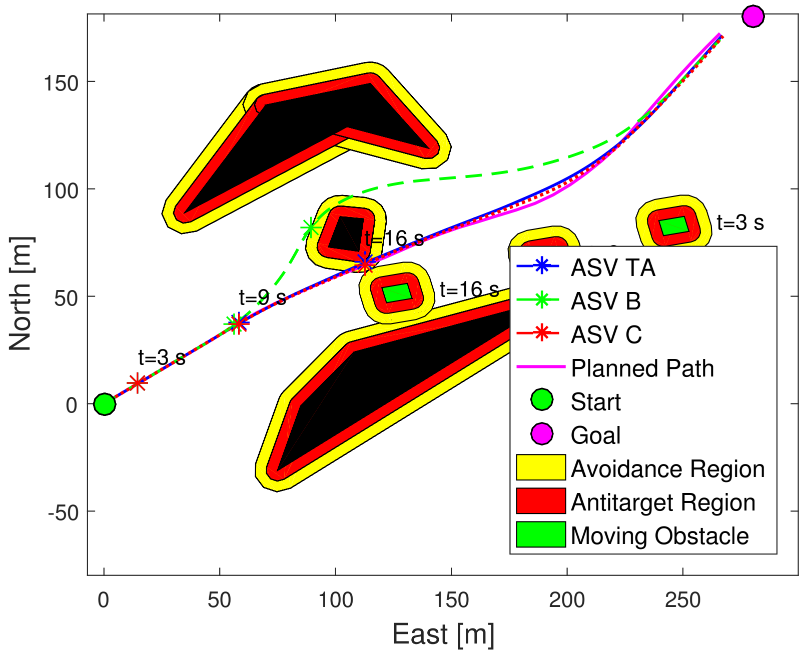
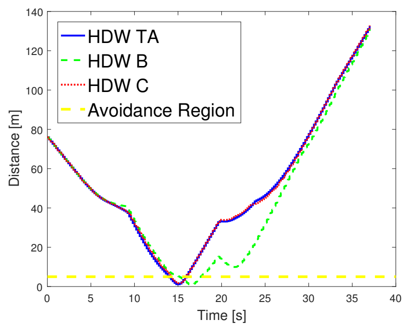
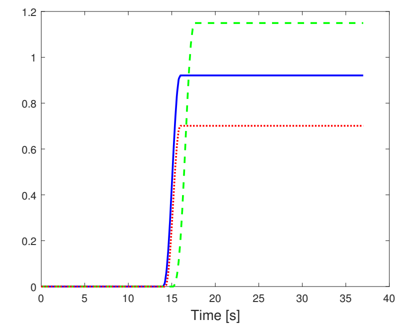


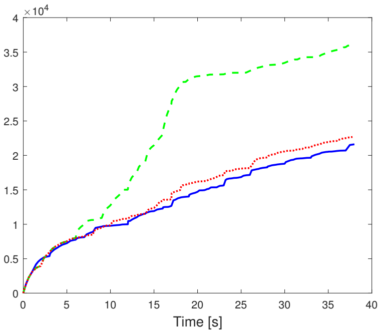
Figure 6.28: Scenario 7 - Hybrid method with RRT + HDW.



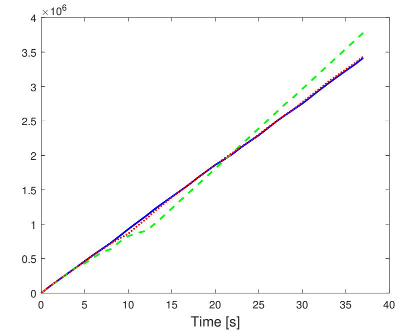
(a) $D_{\mathcal{T}}(t)$



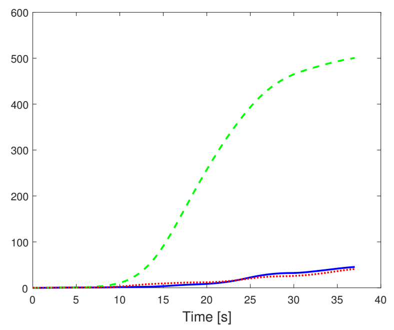
(b) IDI(t)



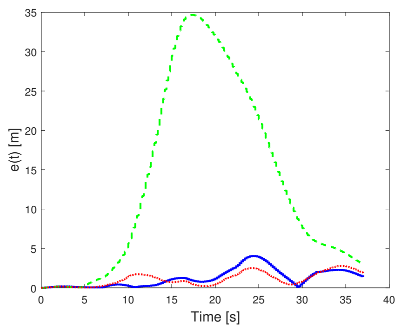
(c) IADC(t)



(d) W(t)



(e) IAE(t)



(f) $e(t)$

Figure 6.29: Scenario 7 - Performance metrics. The blue, green and red lines denote the HDW algorithms TA, B, and C, respectively.

Scenario 8

In Scenario 8, the RRT algorithm is unaware of the moving obstacle, and will therefore lead straight into it. This forces the reactive algorithm to lead the ASV away from the planned trajectory and makes HDW B and HDW C unable to follow it. Even though the ASV is lead away from the planned trajectory, it is able to reach the goal. If the path around the obstacle had been too long, the algorithm would maybe need to create a new planned trajectory by running the RRT algorithm an additional time.

While HDW B and HDW C are able to avoid the obstacle, HDW TA only cares about following admissible paths and the planned trajectory, which in this case is not sufficient to avoid a collision.

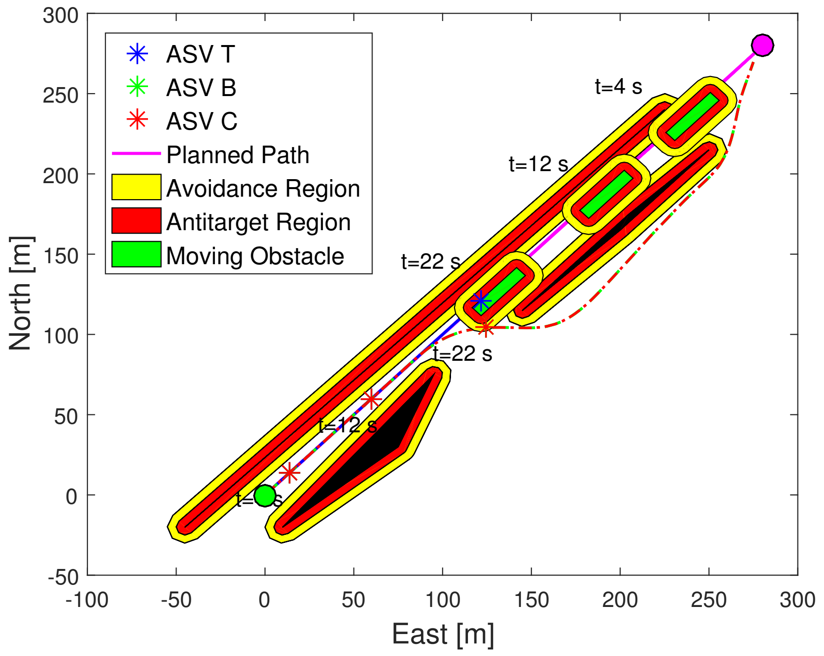
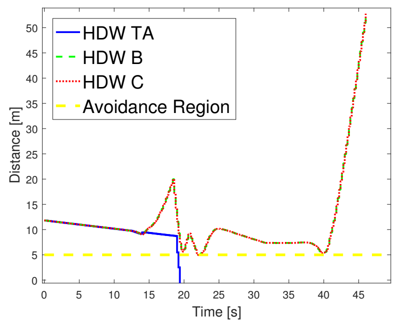
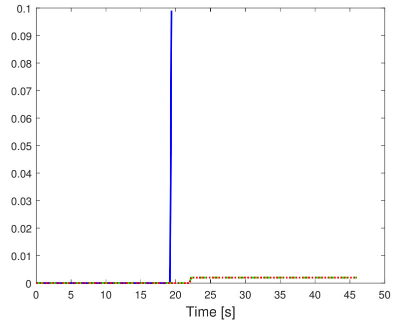


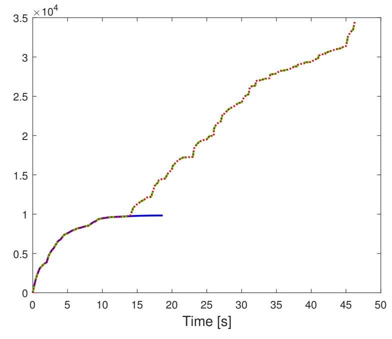
Figure 6.30: Scenario 8 - Hybrid method with RRT + HDW.



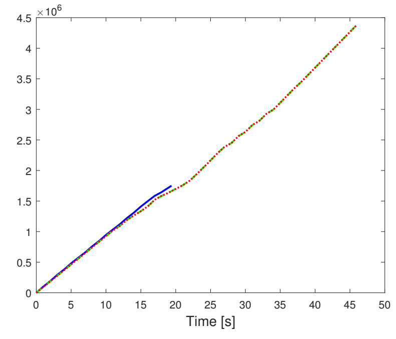
(a) $D_{\mathcal{T}}(t)$



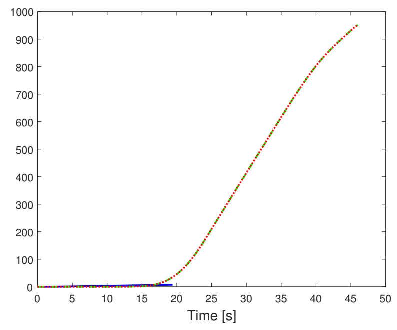
(b) IDI(t)



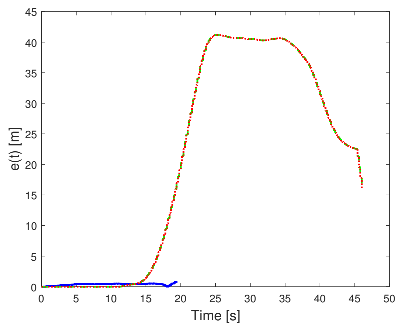
(c) IADC(t)



(d) W(t)



(e) IAE(t)



(f) $e(t)$

Figure 6.31: Scenario 8 - Performance metrics. HDW TA stops at 22 seconds due to collision. The blue, green and red lines denote the HDW algorithms TA, B, and C, respectively.

Scenario 9

This scenario demonstrates how the ASV easily can be trapped in a local minima and stopping when only using a reactive algorithm. The ASV using HDW + RRT algorithm is easily avoiding the local minima and reaches the goal, while the ASV using only DW algorithm gets stuck as seen in Figure 6.32. These results are expected based on the properties of deliberate and reactive methods but are included to underline the need for a hybrid method.

Considering this scenario is solely used to illustrate a reactive method being method getting stuck in a local minima, no further performance evaluations are done.

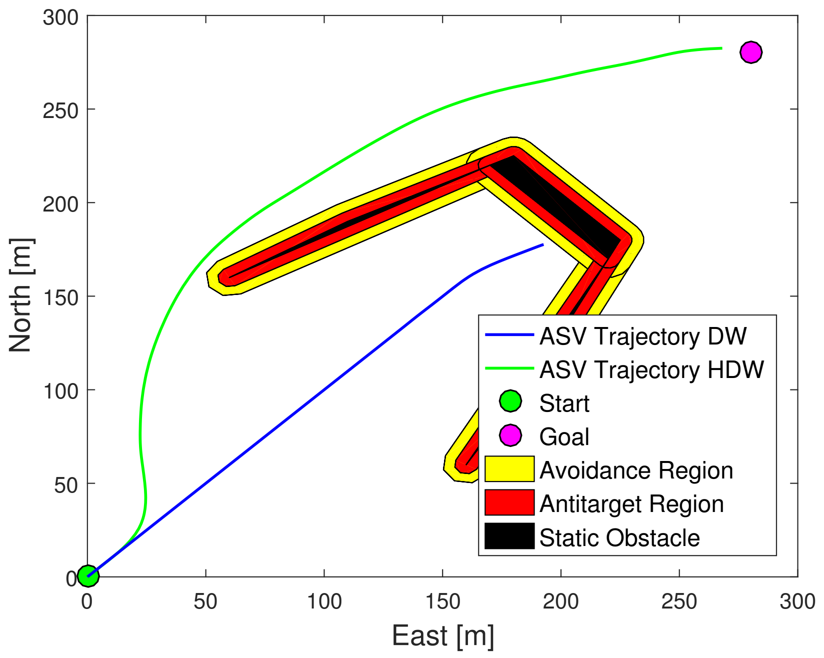


Figure 6.32: Scenario 9 - DW vs HDW + RRT.

Summary of Results

The simulation results presented in this section are evaluated based on the performance metrics described in sections 6.1.1 and 6.2.1, and by qualitatively evaluating the ASV trajectories. A complete overview of the performance metrics at the end of each simulation is listed in Table 6.5. Although a hybrid COLAV method controls the ASVs in the simulations, the evaluation focus is on the HDW ability to avoid obstacles and to follow the planned trajectory generated by the RRT algorithm. Even though the RRT method is not the main focus of the thesis, it is worth to notice that the algorithm generates trajectories which in most cases are feasible for the ASV.

Table 6.5: Performance metric values of the HDW algorithms TA, B and C at simulation end time T. If an ASV has a higher performance metric than the other ASVs in a scenario, the values are bold.

Scenario	**HDW**	$\boldsymbol{D_{min}}$	**IDI**	**IADC**	$\boldsymbol{W}$	**IAE**
1	TA	**30.6**	0	$\mathbf{2.2 \cdot 10^4}$	$4.4 \cdot 10^6$	43.8
	B	28.4	0	$2.6 \cdot 10^4$	$4.4 \cdot 10^6$	**38.8**
	C	28.4	0	$2.6 \cdot 10^4$	$4.4 \cdot 10^6$	**38.8**
2	TA	-	-	-	-	-
	B	5.5	0	$4.9 \cdot 10^4$	$5.3 \cdot 10^6$	108.1
	C	5.5	0	$4.9 \cdot 10^4$	$5.3 \cdot 10^6$	108.1
3	TA	18.6	0	$4.8 \cdot 10^4$	$5.0 \cdot 10^6$	97.7
	B	**18.8**	0	$\mathbf{3.6 \cdot 10^4}$	$5.0 \cdot 10^6$	**94.6**
	C	**18.8**	0	$\mathbf{3.6 \cdot 10^4}$	$5.0 \cdot 10^6$	**94.6**
4	TA	3.4	0.3	$\mathbf{4.2 \cdot 10^4}$	$4.3 \cdot 10^6$	77.8
	B	**4.2**	**0.1**	$5.0 \cdot 10^4$	$4.3 \cdot 10^6$	**72.7**
	C	**4.2**	**0.1**	$5.0 \cdot 10^4$	$4.3 \cdot 10^6$	**72.7**
5	TA	4.9	0.0	$\mathbf{1.8 \cdot 10^4}$	$4.1 \cdot 10^6$	18.4
	B	**5.4**	0	$2.6 \cdot 10^4$	$4.1 \cdot 10^6$	26.9
	C	5.3	0	$2.5 \cdot 10^4$	$4.1 \cdot 10^6$	**17.6**
6	TA	2.4	3.5	$\mathbf{4.3 \cdot 10^4}$	$4.6 \cdot 10^6$	**39.4**
	B	**2.7**	**2.0**	$5.3 \cdot 10^4$	$4.6 \cdot 10^6$	69.0
	C	**2.7**	**2.0**	$4.5 \cdot 10^4$	$4.6 \cdot 10^6$	61.8
7	TA	3.3	0.9	$\mathbf{2.2 \cdot 10^4}$	$\mathbf{3.4 \cdot 10^6}$	47.0
	B	0.9	1.2	$3.6 \cdot 10^4$	$3.8 \cdot 10^6$	504
	C	**4.0**	**0.7**	$2.3 \cdot 10^4$	$\mathbf{3.4 \cdot 10^6}$	**44.5**
8	TA	-	-	-	-	-
	B	5.0	0.0	$3.5 \cdot 10^4$	$4.4 \cdot 10^6$	952
	C	5.0	0.0	$3.5 \cdot 10^4$	$4.4 \cdot 10^6$	952

Based on the performance metrics and qualitatively evaluation, Table 6.6 is formed to give an overview of the simulation results where every algorithm in every scenario is rated as either collided, bad, good or perfect. A result is marked as collided if the ASV enters the antitarget region $\mathcal{T}$. A path leading to the goal, but barely avoiding a collision is marked as bad, while good and perfect denotes increasing performances in the same order. Since the RRT algorithm may generate infeasible trajectories, the HDW algorithm may perform well in some scenarios even though the IAE value is high. In addition, some scenarios make it necessary to leave the planned trajectory to avoid collisions. Hence, not following the planned trajectory may in some scenarios be the optimal behaviour.

Table 6.6: Overview of simulation results for the HDW algorithms. Results are rated as either collided, bad, good or perfect.

Scenario	**HDW TA**	**HDW B**	**HDW C**
1	Perfect	Perfect	Perfect
2	Collided	Good	Good
3	Perfect	Perfect	Perfect
4 mov	Good	Perfect	Perfect
5	Perfect	Perfect	Perfect
6 mov	Good	Perfect	Perfect
7 mov	Good	Bad	Perfect
8 mov	Collided	Perfect	Perfect

HDW TA ignores the avoidance region and focuses only on tracking the planned trajectory generated by the RRT algorithm. From the HDW TA results, we can see that the HDW function works well whenever the planned trajectory keeps a fair distance from obstacles. Hence, using HDW as a trajectory tracking method works well when not travelling close to obstacles.

By including the results of HDW B and HDW C, it is clear that the HDW algorithm is able to both follow the planned trajectory and avoid collisions at the same time. However, to avoid collisions, the ASV may be forced to leave the planned trajectory, but the HDW algorithm is able to catch up with the trajectory once it passes the obstacle. Scenario 5 illustrates well how HDW B and C stay clear of the avoidance region at the cost of slightly moving away from the planned trajectory. The results in Scenario 8 indicate that the HDW algorithm can avoid obstacles which the RRT algorithm is unaware of. Furthermore, this result shows how the ASV is able to reach the goal even when leaving the planned trajectory.

The HDW C algorithm scores equally or better than HDW B for the performance metrics IDI, IADC, W and IAE. There is no scenario where HDW C causes the ASV to enter further into the avoidance region than HDW B. In addition, HDW B returns a bad trajectory in Scenario 7, where it is close to colliding.

To sum up, the HDW algorithm performs well in collision avoidance and is able to track the planned trajectory when a feasible trajectory is generated by the RRT algorithm. As in Section 6.1.2, Algorithm C performs consistently well and outperforms the other methods in most scenarios.

Chapter 7

Conclusion and Suggestions for Future Work

The DW Algorithm A adapts the DW algorithm for use on vehicles with second order nonholonomic constraints and time varying acceleration limits. The algorithm greatly improves the ability to avoid collisions but the avoidance region introduces some new weaknesses. The Algorithm A distance function is disabled when the vehicle resides inside the avoidance region, which makes the algorithm unable to evaluate the ASV distance to surrounding obstacles. Furthermore, Algorithm A is unprovoked to leave the avoidance region if entered. The function is unable to consider trajectories past the region entrance when the vehicle is not inside of it, which may rule out good trajectory options. In addition to these drawbacks, Algorithm A has the general weaknesses of reactive algorithms, which can be accounted for by implementing the algorithm in a hybrid COLAV method.

Two new algorithms, Algorithm B and Algorithm C, are introduced in this thesis with the intention to handle the weaknesses of Algorithm A. Algorithm B is designed to behave identically to Algorithm A when the ASV is far from the avoidance region but differs when being close to or inside the avoidance region. If Algorithm B is close to or inside the avoidance region, it values how fast a trajectory leads out of the avoidance region instead of considering the time before entering the region. As intended, algorithms A and B behave identically in the simulation result when the ASV is far from the avoidance region. When the ASV is close to, or inside the avoidance region, Algorithm B is able to find the closest exit, while Algorithm A will choose the admissible trajectory that leads closest to the goal. This results in Algorithm A often colliding or being close to colliding, while Algorithm B behaves more rationally. Based on these results, it is clear that Algorithm B is superior to Algorithm A in the tested scenarios.

Algorithm B has to be close to the avoidance region to consider what lies beyond the entrance of the region, which may rule out good trajectory options. The distance function of Algorithm C, however, is based partially on the distance to the avoidance region and partially on how much of the predicted trajectory resides inside the avoidance region. Consequently, Algorithm C is able to consider what lies beyond the avoidance region entrance and is less hesitant to cross through the avoidance region if this is necessary, or the optimal solution. Algorithm C performs consistently equally as good or better than Algorithm B in the simulation results. Since Algorithm C is able to early consider trajectories through the avoidance region, the algorithm has more options when trying to avoid a dangerous situation. Based on the simulation results, Algorithm C is superior to algorithms A and B, and greatly reduces the drawbacks of the avoidance region.

In addition to the simulation study presented in this thesis, Algorithm C was used in a full-scale test in Trondheim between 15th and 19th of May 2017 on the Telemetron ASV described in [14]. Both the original DW algorithm and DW Algorithm C were used, where Algorithm C caused the best algorithm behaviour among the two. However, the sensor signals and radar signals were noisy, which caused problems for both the DW algorithms. Hence, no conclusion was made, even though Algorithm C returned better behaviour than the original DW algorithm.

To adapt the DW algorithm for use in hybrid COLAV methods, the HDW algorithm is introduced. The new algorithm uses the distance function of either algorithm A, B or C. The algorithm receives a planned trajectory from a deliberate method, e.g. the RRT algorithm which is used in the simulations. An alignment function is used in the HDW algorithm, where the predicted trajectories of velocity pairs are compared to the planned trajectory. Furthermore, the HDW algorithm use predicted trajectories where the desired surge speed and yaw rate change along the trajectory. This increases the computational load significantly, but are still doable when using two velocity pairs for every trajectory. By evaluating the distance function and the alignment function, the HDW is able to both track the trajectory and avoid obstacles. Hence, the HDW algorithm functions as both the reactive method of the hybrid COLAV method but also as the interface between the reactive and deliberate method. Furthermore, the new algorithm reduces the number of tuning parameters from three to one.

The HDW method is simulated with distance function B, distance function C and as a pure trajectory tracker, denoted HDW B, HDW C and HDW TA, respectively. As for the reactive methods, distance function C causes the algorithms to be less hesitant to go through the avoidance region. When the deliberate method generates a trajectory that goes through the avoidance region to avoid a collision, it is important that the HDW algorithm is able to follow the trajectory. Due to this, HDW C scores the best in the simulation results and is the superior among the tested algorithms. Finally, it is concluded that the HDW algorithm is successfully able to track the feasible trajectory given by the deliberate method and avoid collisions in high speed.

The issues of the avoidance region are greatly reduced by the introduction of Algorithm C. Furthermore, introducing HDW C has significantly improved the COLAV performance of the DW algorithm for ASVs. The hybrid COLAV method formed by the RRT and HDW algorithms yields a consistently good ASV behaviour, which performs well in both open sea and when operating closer to obstacles.

The following topics should be investigated for further work:

- Adapt the DW and HDW algorithms to follow COLREGS.
- Improve the deliberate method to generate optimal trajectories that are feasible for the ASV.
- Investigate if modifying distance function C to add costs to the velocity pairs based on how deep inside the avoidance region each part of a trajectory resides improves the DW algorithm.
- Simulate a more complete model of the ASV and other ASV models using different desired surge speeds.
- Investigate the robustness of the algorithm with respect to model uncertainties [13].
- Perform full-scale testing with ASVs.

Bibliography

[1] *Autonomous Surface Vehicles (ASV) Ltd.* URL: `http://www.unmannedsystemstechnology.com/2014/02/asv-launch-revolutionary-oil-field-services-unmanned-surface-vehicle/` (visited on 05/24/2017).

[2] M. R. Benjamin, J. J. Leonard, J. A. Curcio, and P. M. Newman. “A Method for Protocol-Based Collision Avoidance Between Autonomous Marine Surface Craft”. In: *Journal of Field Robotics*. Vol. 23(5), pp. 333–346. Wiley-Blackwell, 2006.

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry*. Springer Nature. Chap. 13. 2008.

[4] A. Bloch, P. Crouch, and J. Marsden. *Nonholonomic Mechanics and Control*. Springer. Chap. 5. 2010.

[5] J. Borenstein and Y. Koren. “The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots”. In: *IEEE Transactions on Robotics and Automation*. Vol. 7(3), pp. 278–288. 1991.

[6] M. S. Branicky, S. M. LaValle, K. Olson, and L. Yang. “Deterministic vs. Probabilistic Roadmaps”. In: *IEEE Transactions on Robotics and Automation*, pp. 1–13. 2002.

[7] M. Breivik and T. Fossen. “Path following for marine surface vessels”. In: *Proceedings of the OTO’04*. Kobe, Japan, 2004.

[8] M. Breivik and T. I. Fossen. *Guidance Laws for Autonomous Underwater Vehicles*. InTech. Chap. 4. 2009.

[9] O. Brock and O. Khatib. “High-speed Navigation Using the Global Dynamic Window Approach”. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Detroit, Michigan, 1999.

[10] *Convention on the International Regulations for Preventing Collisions at Sea, 1972 (COLREGs)*. URL: `http://www.imo.org/en/About/Conventions/ListOfConventions/Pages/COLREG.aspx` (visited on 11/03/2016).

[11] G. L. Dirichlet. “Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen.” ger. In: *Journal für die reine und angewandte Mathematik*. Vol. 40, pp. 209–227. 1850.

[12] B.-O. H. Eriksen, M. Breivik, K. Y. Pettersen, and M. S. Wiig. “A modified dynamic window algorithm for horizontal collision avoidance for AUVs”. In: *Proc. of 2016 IEEE Conference on Control Applications (CCA)*. Buenos Aires, Argentina, Sept. 2016.

[13] B.-O. H. Eriksen. “Horizontal Collision Avoidance for Autonomous Underwater Vehicles”. MA thesis. Trondheim, Norway: Norwegian University of Science and Technology, 2015.

[14] B.-O. H. Eriksen and M. Breivik. “Modeling, Identification and Control of High-Speed ASVs: Theory and Experiments”. In: *Sensing and Control for Autonomous Vehicles: Applications to Land, Water and Air Vehicles*. Lecture Notes in Control and Information Sciences. Springer, 2017.

[15] P. Fiorini and Z. Shiller. “Motion Planning in Dynamic Environments using Velocity Obstacles”. In: *Int. J. Robot. Res.* Vol. 17, pp. 760–772. 1998.

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

[17] D. Fox, W. Burgard, and S. Thrun. “The Dynamic Window Approach to Collision Avoidance”. In: *IEEE Robotics & Automation Magazine*. Vol. 4(1). Institute of Electrical and Electronics Engineers (IEEE), pp. 23–33. 1997.

[18] E. Frazzoli, M. A. Dahleh, and E. Feron. “Real-time motion planning for agile autonomous vehicles”. In: *Journal of Guidance, Control and Dynamics*. Vol. 25(1), pp. 116–129. 2002.

[19] P. Hart, N. Nilsson, and B. Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics*. Vol. 4(2). Institute of Electrical and Electronics Engineers (IEEE), pp. 100–107. 1968.

[20] J. P. Hespanha. *Linear System Theory*. Princeton University Press. New Jersey, 2009.

[21] A. Isidori. *Nonlinear Control Systems*. 2nd edition. Berlin: Springer-Verlag, p. 23. 1989.

[22] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces”. In: *Proceedings of IEEE Transactions on Robotics and Automation*. Vol. 12(4), pp. 556–580. 1996.

[23] O. Khatib. “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots”. In: *The International Journal of Robotics Research*. Vol. 5(1). SAGE Publications, pp. 90–98. 1986.

[24] 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. Sacramento, California, 1991.

[25] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger. “Safe Maritime Autonomous Navigation With COLREGS, Using Velocity Obstacles”. In: *IEEE Journal of Oceanic Engineering*. Vol. 39(1). Institute of Electrical and Electronics Engineers (IEEE), pp. 110–119. 2014.

[26] J. Larson, M. Bruch, and J. Ebken. “Autonomous Navigation and Obstacle Avoidance for Unmanned Surface Vehicles”. In: *Proceedings of SPIE - The International Society for Optical Engineering*. Orlando,Florida, 2006.

[27] S. M. LaValle. *Rapidly-exploring random trees: A new tool for path planning*. Tech. rep. Computer Science Dept., Iowa State University, 1998.

[28] Ø. A. G. Loe. “Collision Avoidance for Unmanned Surface Vehicles”. MA thesis. Trondheim, Norway: Norwegian University of Science and Technology, 2008.

[29] P. Ögren and N. Leonard. “A Convergent Dynamic Window Approach to Obstacle Avoidance”. In: *IEEE Transactions on Robotics*. Vol. 21(2). Institute of Electrical and Electronics Engineers (IEEE), pp. 188–195. Barcelona, Spain, 2005.

[30] G. Oriolo and Y. Nakamura. “Control of Mechanical Systems With Second-order Nonholonomic Constraints: Underactuated Manipulators”. In: *Proc. of the 30th IEEE Conference on Decision and Control*. Brighton, England, 1991.

[31] M. Seder, K. Macek, and I. Petrovic. “An integrated apporach to real-time mobile robot control in partially known indoor environments”. In: *31st Annual Conference of IEEE Industrial Electornics Society*, pp. 1785–1790. Raleigh, North Carolina, 2005.

[32] E. Serigstad. *Collision avoidance for ASVs using Dynamic Window*. Project report, Norwegian University of Science and Technology: Trondheim, Norway, 2016.

[33] R. Skjetne, T. I. Fossen, and P. V. Kokotović. “Robust Output Maneuvering for a Class of Nonlinear Systems”. In: *Automatica*. Vol. 40(3). Elsevier BV, pp. 373–383. 2004.

[34] SNAME. *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid*. Tech. rep. New York. USA, 1950.

[35] M. E. N. Sørensen and M. Breivik. “Comparing Nonlinear Adaptive Motion Controllers for Marine Surface Vessels”. In: *Proceedings of the 10th IFAC Conference on Manoeuvring and Control of Marine Craft*. Vol. 48(16). Elsevier BV, pp. 291–298. Copenhagen, Denmark, 2015.

[36] M. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons, Inc., 2006.

[37] C. S. Tan, R. Sutton, and J. Chudley. “An incremental stochastic motion planning technique for autonomous underwater vehicles”. In: *Proceedings of IFAC Control Applications in Marine Systems Conference*, pp. 483–488. Ancona, Italy, 2004.

[38] *Viknes Båt og Service AS*. URL: `http://viknes.no/modell/viknes-830/` (visited on 05/10/2017).

[39] K. Wichlund, O. Sørdalen, and O. Egeland. *Control Properties of Underactuated Vehicles*. In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Nagoya, Japan, 1995.

[40] S. Zaheer, J. M, and T. Gulrez. “Performance Analysis of Path Planning Techniques for Autonomous Mobile Robots”. In: *Proceedings of 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. Coimbatore, India, 2015.