Master's thesis

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

Amalie Heiberg

COLREG-Compliance for Autonomous Surface Vehicles using Deep Reinforcement Learning

Two approaches

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
July 2020

NTNU
Norwegian University of Science and Technology

Amalie Heiberg

COLREG-Compliance for Autonomous Surface Vehicles using Deep Reinforcement Learning

Two approaches

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
July 2020

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

NTNU
Norwegian University of
Science and Technology

Preface

This thesis concludes my Master's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology, and is the fruit of work conducted during the spring of 2020. Although it is not a direct extension of the project thesis carried out in the autumn of 2019, the background knowledge and experience with reinforcement learning gained has been utilised.

The topic of the work is COLREG-compliant collision avoidance applied to autonomous surface vehicles, and is built on those of Haakon Robinson and Eivind Meyer, who developed the framework serving as an excellent starting point. Their framework, as well as my continuation, is developed in the Python programming language and makes extensive use of the OpenAI Gym [1] and Stable Baselines [2] libraries. The former is a toolkit for developing reinforcement learning algorithms, and the latter provides full implementations of such algorithms. In addition, the NumPy package [3] for scientific computing was used for numerical computations, and the Shapely package [4] for computational geometry was used to aid geometric representation and manipulation of static and dynamic obstacles.

In order to explore COLREG-compliance in a reinforcement learning framework, two approaches are taken and compared. The first is a qualitative one, directly making use of the sensor input available via the virtual sensor suite. I'd like to thank Eivind Meyer for serving as a discussion partner throughout this stage of the work, which eventually lead to our collaboration on a research paper [5]. This is contrasted with a risk-based approach, which to a larger extent employs empirical data and expert knowledge.

Finally, I would like to thank my supervisor Professor Adil Rasheed for his tireless support and inspirational attitude, which has been especially appreciated in these unusual times.

Amalie Heiberg,
Trondheim, July 15, 2020

Abstract

Autonomous systems are becoming ubiquitous, and are now also gaining momentum within the marine sector. Since the electrification of transport is happening at the same time, the envisioned autonomous vessels promise reduced environmental impact, lower costs, and higher efficiency. Although close monitoring is still required to ensure safety, the ultimate goal would be total autonomy. One of the major hurdles is the development of a control system versatile enough to handle all possible weather and encounter situations, that is also robust and reliable. Additionally, the International Regulations for Preventing Collisions at Sea (COLREGs) must be followed for successful interaction with human sailors. Since the COLREGs were written for the human mind to interpret, they are written in ambiguous prose and therefore not machine readable or verifiable.

Due to these challenges and the wide variety of situations to be tackled, classical model-based approaches prove complicated to implement and computationally heavy. Within the field of artificial intelligence, deep reinforcement learning (DRL) has shown great potential for a wide range of applications. Its model-free and self-learning nature makes it a promising candidate for autonomous vessels. In this thesis, two ways of incorporating the COLREGs into a DRL-based path following and obstacle avoidance system are explored. First, the direct usage of sensor data combined with intuition is looked into. Then, a system based on readily available theory of collision risk is developed.

Both of the approaches provide good results in testing scenarios, adhering to the COLREG rules relevant to a single-agent environment – rules 14-16. This means that in addition to achieving excellent path following and collision avoidance performance in the face of static obstacles, the DRL agent adhered the implemented COLREGs in situations where the desired behaviour was clearly defined. In both cases, it was shown that a modular approach to reward function design works well in DRL applications with multiple objectives.

The successful inclusion of key COLREG rules into a well-functioning path following and collision avoidance system is testament to the potential of DRL in autonomous vessels.

Sammendrag

Bruken av og forskning innen autonome systemer har økt kraftig i senere år, inkludert i marin sektor. Ettersom transportsektoren samtidig gjennomgår en omfattende elektrifisering, lover autonom skipsfart ikke bare reduserte kostnader gjennom nedbemanning og mer effektiv drift, men også reduserte utslipp. Helautonomi kan derfor sies å være et fremtidig mål, selv om det i dag kreves konstant monitorering av delvis autonome skip. Et av de største hindrene for å nå dette målet er utviklingen av et robust og pålitelig kontrollsystem som er i stand til å takle alle mulige situasjoner og vær. Videre er det essensielt at alle skip følger internasjonale regler for kollisjonsunngåelse på havet (engelsk forkortelse: COLREGs), slik at samarbeidet med kapteiner og andre mennesker er trygt. Siden COLREGs ble skrevet for mennesker, er de ofte formulert på tvetydig vis, og dermed ikke lett overførbare til eller verifiserbare i en digital kontekst.

Grunnet disse utfordringene er det teknisk krevende å nå målet kun ved bruk av klassiske og modell-baserte metoder. Kunstig intelligens kan approksimere beslutningsmodeller, og virker derfor lovende. Forsterkende læring (engelsk: reinforcement learning) har vist et spesielt stort potensiale i et bredt spekter av applikasjoner, inkludert de som krever kontinuerlig tilstands- og handlingsrom. Siden forsterkende læring i tillegg er en selvlærende og modellfri metode er det en spesielt god kandidat for autonome skip. I denne masteroppgaven undersøkes potensialet for å flette COLREGs inn i en kontroller basert på dyp forsterkende læring (engelsk forkortelse: DRL). For å oppnå dette sammenliknes en kvalitativ og en risiko-basert metode.

Begge metodene fører til gode resultater i testscenarioer, og følger COLREG-regler relevante i et miljø med én aktiv agent (regler 14-16). Dette betyr at, i tillegg til å oppnå svært god stifølging og kollisjonsunngåelse i møte med statiske objekter, var agentene i stand til å forholde seg til de implementerte COLREG-reglene. I begge tilfeller var det tydelig at en modulær funksjon for belønning fungerer godt i applikasjoner hvor agenten skal oppnå ulike konkurrerende mål.

Den vellykkede inkluderingen av viktige COLREG-regler i et DRL-basert system for stifølging og kollisjonsunngåelse vitner om at DRL er gunstig for autonom navigasjon på havet.

Table of Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AI	Artificial intelligence
AIS	Automatic identification system
ANN	Artificial neural network
APF	Artificial Potential Field
ARPA	Automatic radar plotting aid
CAS	Collision alert system
CNN	Convolutional neural network
COLAV	Collision avoidance
COLREGs	International Regulations for Preventing Collisions at Sea
CPA	Closest point of approach
CRI	Collision risk index
CTE	Cross-track error
DCPA	Distance to closest point of approach
DDPG	Deep deterministic policy gradient
DL	Deep learning
DOF	Degrees of freedom

DP	Dynamic programming
DQN	Deep Q-network
DRL	Deep reinforcement learning
DW	Dynamic Window
GA	Genetic algorithm
GNC	Guidance, navigation and control
IvP	Interval programming
KL	Kullback–Leibler
LOS	Line-of-sight
LSTM	Long short-term memory
MDP	Markov decision process
ML	Machine learning
MPC	Model predictive control
MSE	Mean square error
NED	North, East, Down
NN	Neural network
OS	Ownship
PID	Proportional–integral–derivative
POMDP	Partially observable Markov decision process
PPO	Proximal policy optimisation
RL	Reinforcement learning
RNN	Recurrent neural network
SARSA	State-action-reward-state-action
TCPA	Time to closest point of approach
TD	Temporal difference
TRPO	Trust region policy optimisation
TS	Target ship

VO	Velocity Obstacles

Symbols

α	Neural network learning rate
α_p	Path tangential angle
β_c	Crab angle
$\boldsymbol{\eta}$	Pose vector
$\boldsymbol{\nu}$	Velocity vector
$\boldsymbol{\omega}$	Angular velocity vector
$\boldsymbol{\tau}$	Force vector
$\boldsymbol{\Theta}$	Euler angles
$\boldsymbol{\theta}$	Parameters (weights and biases) of neural network
$\boldsymbol{b}$	Bias vector in neural network
$\boldsymbol{f}$	Linear force vector
$\boldsymbol{m}$	Moment vector
$\boldsymbol{p}$	Position vector
$\boldsymbol{p}_d$	Look-ahead reference point
$\boldsymbol{p}_e$	Path error vector
$\boldsymbol{v}$	Linear velocity vector
$\boldsymbol{W}$	Weight matrix in neural network
χ	Course
χ_r	Relative course between ownship and target vessel
Δ_{LA}	Look-ahead distance
γ	Neural network discount factor
$\mathcal{A}$	Set of possible actions
$\mathcal{R}(s, a)$	Reward function
$\mathcal{S}$	Set of possible states
$\mathcal{T}$	Transition function

$\mathcal{T}(s, a, s')$	Transition function
ψ	Heading
θ_T	Relative bearing of target ship
$A(s, a)$	Advantage function
$L(\cdot)$	Reinforcement learning objective function
L_{pp}	Vessel length
$Q(s, a)$	Estimated action-value function
$q(s, a)$	Action-value function
R	Absolute distance between ownship and target vessel
U	Horizontal-plane speed
$V(s)$	Estimated value function
$v(s)$	Value function
V_r	Relative speed of target ship

Subscripts and superscripts

$*$	Optimality property
$\pi(\cdot)$	Stochastic policy
b	Coordinate in BODY frame
d	Desired value
e	Error
k	Current waypoint
n	Coordinate in NED frame
r	Relative value
t	Time step

Chapter 1

Introduction

In order for autonomous surface vehicles to reach their full operational potential, adherence to the International Regulations for Preventing Collisions at Sea (COLREGs) is essential. However, the incorporation of COLREGs into autonomous navigation has not been sufficiently researched. This thesis explores two routes to autonomous COLREG-compliance in a framework based on deep reinforcement learning.

1.1 Motivation

Over the last few years, the promising idea of autonomous ships has gained traction through projects such as ReVolt by DNV GL [6] and Yara Birkeland by KONGSBERG and Yara [7]. In addition, research into autonomous ships is increasingly being incentivised through funding bodies that recognise the potential benefits of autonomy at sea. A notable example of this is the EU-funded four-year project Autoship Horizon 2020, which seeks to speed-up the transition towards autonomous ships in the EU [8]. For the first time in history, the promise of lower emissions, higher efficiency and fewer accidents via autonomy is becoming tangible.

Human error is a leading cause of accidents on the road [9] [10], and reports show that accidents at sea are no different. According to the Annual Overview of Marine Casualties and Incidents published by the European Maritime Safety Agency

(EMSA), human error was attributed to over 50% of accidental events in the period 2011-17 [11]. In addition to reducing accidents and thereby fatalities, damage to the environment and costs, autonomous marine operations allow for optimised route planning. This can be done with respect to time spent, or fuel costs. It should also be mentioned that autonomous ships can move cargo from the road to the sea, which could lead to less trafficked roads. For instance, the autonomous container ship Yara Birkeland is expected to reduce the amount of trips made by diesel trucks by 40,000 a year after its launch in 2020 [12]. Together with the widespread electrification that is taking place, reduced air pollution is another likely and desirable effect.

An overall reduction of errors as a result of introducing autonomy depends on the development of robust and reliable systems, which is no trivial task. For autonomous navigation at sea, the vessel's control system must deal appropriately with a wide range of situations not only depending on the position of the ownship (OS) and other ships within a certain radius, but also on environmental factors such as wind, ocean currents, and waves. Another crucial element is detection, classification and tracking of objects, which might in itself be challenging in certain weather conditions. As described more thoroughly in the literature review, which follows in Section 1.2, the currently proposed solutions generally make important simplifications and assumptions. Low-level controllers, or autopilots, are already commercially available, but more research on higher level path planning and collision avoidance is needed to ensure safe autonomous navigation in real situations. For collision avoidance, COLREG-compliance is crucial to ensure safety when encountering other vessels.

Due to the complex nature of autonomy at sea, classical model-based methods may be challenging to implement for full autonomy. Since modern AI methods have the advantage of learning, however, such methods could prove useful for model approximation. Supervised learning approaches are in theory very powerful, but are limited by their dependency on good and labelled training data. Reinforcement learning (RL) remedies this by producing data iteratively as the agent, in this case a marine vessel, interacts with its environment and records the outcomes of actions taken.

Motivated to start bridging the gap between state-of-the-art RL and the challenges faced by real vessels, this thesis aims to incorporate the COLREGs into an autonomous path following and collision avoidance system based on deep reinforcement learning (DRL). Two distinct approaches are explored:

1. Based on qualitative methods
2. Based on measures of collision risk and empirical knowledge

The goal is to compare these approaches and achieve a system that generalises well to a wide range of situations, without being computationally expensive in real-time.

1.2 Literature review

In the past, collision alert systems (CAS) were developed to aid the captain and crew on board a vessel. These systems were largely extensions of sensors, presenting the sensor data to the user in a helpful manner. Examples of CAS systems are Automatic Radar Plotting Aid (ARPA) and Automatic Identification System (AIS) (compared in [13]), which are routinely used for collision risk evaluation [14]. As we are moving into the fourth industrial revolution, solutions such as digital twins and remote sensing are making their ways into the maritime industry [15]. Decision-making is thus gradually being taken from the cognitive realm and into the digital domain, and the need for highly robust and flexible guidance, navigation and control (GNC) systems is growing. Since COLAV systems are responsible for one of the most safety-critical aspects of a vessel's operation, an integrous COLAV system is required for any GNC system operating in a dynamic environment [16]. To reach full autonomy at sea, the development of reliable and transparent COLAV systems is therefore crucial.

Before autonomous vessels became a possibility, the International Regulations for Preventing Collisions at Sea (COLREGs) were formulated to prevent collisions between two or more vessels [17]. Although technological advancement has been great since their publication in 1972, COLREG-compliance for autonomous vessels is still understudied. One of the main challenges is that the COLREGs were written for humans to interpret, and must therefore be translated to a machine readable and verifiable format. Another potential challenge is the indirect communication that occurs when two vessels meet in a situation with high risk of collision. For instance, for communication purposes, the COLREGs require relatively sharp manoeuvres whenever a high-risk situation is encountered. However, from an the point of view of a system that bases its decisions on energy efficiency, or even risk of collision, this may not be the optimal behaviour. Hence, the human-machine interface should be given sufficient attention, so that the autonomous vehicle behaves in a way that can be appropriately discerned by a human.

In addition to the challenges inherent to the COLREGs, autonomous collision

avoidance can be demanding due to the complex dynamics of ships, varying speeds, and changing environmental conditions [18]. The majority of the proposed solutions for autonomy make assumptions that do not represent reality. Examples of such assumptions are constant speed of the OS or other ships, good weather conditions, or that the system only operates while the ship is at open sea. It is clear that an adequate autonomous vessel must be able to deal with all the situations the current fleet handles. For instance, given sufficient situational awareness, a full-fledged autonomous COLAV system should be expected to handle situations involving all sorts of moving and stationary objects, from container ships to kayaks. For generalisation, the system must be able to track a high number of objects simultaneously, and perform well in congested waters.

Before embarking on the development of a COLREG-compliant COLAV system based on DRL, it is useful to review the currently proposed solutions. A plethora of COLAV algorithms and architectures for autonomous control have been and are being researched, and there are many ways to distinguish between these.

1.2.1 Broad comparison of COLAV systems

Broadly speaking, COLAV systems can be classified from three perspectives:

1. **Classical and soft systems.** One way to look at COLAV systems is through the lens of so-called classical and soft systems, as described by Statheros et al. [19]. Classical systems are based on mathematical models and logic, and thus assume that an optimum can be found analytically or numerically. As a result, proof of convergence can normally be found, which is one of the reasons the classical approach is widely used in industry. For collision avoidance, and many other applications, model predictive control (MPC) is popular. As shown by Johansen et al. [20] and Eriksen [21], MPC can be used to develop a COLAV system compliant with the main rules of COLREGs. However, this method is simulation-based, and would become computationally expensive if it were to consider a high number of control behaviours at each interval. Another challenge pointed out in [20] is the dependency of performance on parameter tuning, which is likely to be time-consuming. However, the method is powerful due to its ability to formalise for instance physical constraints and risk measures through cost functions. MPC can also be applied to nonlinear systems with uncertain environmental disturbances, as done by Soloperto et al. [22].

   MPC is popular, but several other classical methods can also be used for col-

lision avoidance. Two of these, which have been applied to marine vessels, are the Velocity Obstacle (VO) and Interval Programming (IvP) methods. The VO method was first proposed by Fiorini and Shiller in 1998 [23], and works by creating velocity obstacles: artificial obstacles representing the velocities that would result in collisions. This method assumes constant velocity for both the OS and other vessels. Kuwata et al. [24] showed that maritime navigation can be done according to the COLREGs when using the VO method. Interval programming has also been shown to successfully produce COLAV systems adhering to the COLREGs [25] [26]. In the collision avoidance application, a multi-objective optimisation approach is taken, where weighted IvP functions are used as objective functions. Dynamic Window (DW) is another optimisation-based method that has been researched for marine applications [27]. The strength of DW can be found in its focus on fast dynamics, resulting from a reduction of the search space to velocities reachable within a short time interval [28].

In contrast to classical systems, soft systems assume that the problem at hand is not readily quantified, and are based on artificial intelligence (AI). A main group of methods adhering to the soft approach is heuristics, which are experience-based methods for finding an acceptable solution to a problem. The A* heuristic, introduced by Hart, Nilsson, and Raphael in 1968 [29], might be the most well-known and widely used. It is a greedy algorithm for finding the shortest distance employing a heuristic for the distance of each point to the goal. This means that it is often used for high-level path and trajectory planning, as was done in [21]. Another widely known heuristic technique is the Artificial Potential Field (APF) method introduced by Khatib in 1985 [30]. Simply put, it works by creating attractive and repulsive artificial fields, on which gradient descent can be done. A main challenge with the APF method is its tendency to get trapped in local minima. In [31] it was shown that this can be remedied for collision avoidance when using an adaptive version of APF. A third well-known heuristic is the genetic algorithm (GA), which is based on evolutionary theory. Such an approach is taken in [32], where a genetic algorithm is used for trajectory planning in an environment with static and dynamic obstacles. APF and GAs are only two examples of the wide variety of heuristic methods that can be applied to collision avoidance, and each method exhibits different strengths and weaknesses. For example, [33] showed that Distributed Tabu Search, a metaheuristic method, can be used for collision avoidance in highly congested areas.

Another set of soft systems is that of machine learning (ML). ML techniques such as deep learning (DL) and reinforcement learning (RL) have gotten a lot of attention in the context of autonomous systems. An advantage of ML techniques is their data-driven nature, which means that they do not require mathematical models of the vessel of the environment. They also allow transfer learning, which means that knowledge obtained in one situation can be applied to another. However, only a limited amount of the research has been devoted to autonomous marine vessels, compared to e.g. driverless cars. In [34], a deep convolutional neural network (CNN) is trained for COLREGs compliant collision avoidance for an unmanned surface vehicle. This method is based on image recognition, and makes use of the visual capabilities of CNNs. Although the results are compelling, an adequate use of CNNs requires well-balanced and sufficient training data. Due to the lack of real data, [34] relies on data collected from a ship simulator game. RL shows great potential, as straight-path following, curved path following and simple collision avoidance by RL has been documented for marine vessel models in [35], [36], and [37] respectively. In these works, the Deep Deterministic Policy Gradient (DDPG) [38] algorithm was employed. In addition, Zhao et al. [39] and Meyer et al. [40] showcased the potency of the continuous actor-critic method Proximal Policy Optimisation (PPO) [41] for multiship collision avoidance.

2. **Deliberative and reactive systems.** Secondly, COLAV systems can be compared via their deliberative and reactive properties [42]. Deliberative systems can be described by "sense-plan-act", as they aim to use information to plan into the future. Reactive systems, on the other hand, can be seen as "sense-act" systems. These systems exhibit tight coupling and low time delay between sensed input and actions. In other words, deliberative and reactive systems are path-planning and low-level control systems, respectively. Some of the classical systems described above are deliberative, while some are reactive. For instance, MPC-based systems are deliberative, whilst IvP, DW, and VO methods are reactive. On the side of soft systems, A* is deliberative, and the APF method is reactive

3. **Modes of communication.** A third way of comparing COLAV systems is by looking at their modes of communication. All vessels above 300 tonnes engaged on international voyages, all cargo ships above 500 tonnes, and all passenger ships are required to carry an AIS [43]. The AIS transmits and receives information about identity, position, course, speed, etc., which can be incorporated into a COLAV system. Such systems can thus enhance the

quality of information about other vessels, but depend on infrastructure and communication with other vehicles. Since one cannot expect complete availability, a ship is usually equipped with exteroceptive sensors such as cameras, lidars, and radars, in addition. For an autonomous COLAV system, it would be beneficial to make use of AIS information, without depending on it.

Several hybrid COLAV systems can and have also been implemented. When combining for instance classical and soft systems, or deliberative and reactive systems, we get hybrid systems. This is done with increasing frequency [44]. Multi-layered systems are also being developed, where each subsystem lies on a spectrum between reactive and deliberative. Such hybrid architectures are able to harvest the strengths of several methods, using each where they perform best. It can be argued that modular and layered hybrid systems are likely to become increasingly popular, due to their intuitive nature. Loe [45] uses a two-layered approach, where deliberation is done by a Rapidly-Exploring Random Tree (RRT) algorithm combined with A*, and the reactive component consists of a modified Dynamic Window algorithm. In [21], the A* deliberative heuristic is combined with a mid-layer and a reactive MPC-based algorithm, forming a three-layered COLAV system. Other layered architectures have been proposed by Casalino et al. [46] and Švec et al. [47].

In summary, a wide range of COLAV systems have been proposed in literature, generally disregarding the COLREGs. At the same time, the increased focus on autonomous systems in later years requires COLREG-compliance for sufficient safety. This gap combined with the promise of DRL for autonomous navigation shapes the objectives of the thesis.

1.3 Objectives

Primary objective: Investigate COLREG-compliance in a path following and collision avoidance system based on deep reinforcement learning through the comparison of two approaches.

Since the COLREGs have been developed to reduce risk of collision, it is interesting to assess the current methods for measuring collision risk from a DRL perspective. At the same time, the design of DRL systems is often done using intuition rather than step-by-step methods, due to the lack of effective methodologies. Comparing a qualitative approach with a risk-based one could therefore provide useful insights. Hence, the two approaches to be implemented and compared are:

- **Qualitative approach:** aims to incorporate the COLREGs into a DRL controller using intuition
- **Risk-based approach:** aims to incorporate the COLREGs into a DRL controller using state-of-the-art collision risk measures

Doing so, the thesis seeks to answer the following **research questions:**

- Which of the approaches implemented shows the largest potential for COLREG-compliant collision avoidance?
- Is deep reinforcement learning suited for COLREG-compliant navigation at sea?

1.4 Outline of report

The thesis is comprised of five chapters, of which the first is this introductory chapter providing the main motivation for carrying out the work, along with a brief literature review of collision avoidance at sea. Moreover, the objectives and contributions are specified. In Chapter 2, the necessary background is presented. The chapter presents the most important concepts relevant to this work, namely marine vessel modelling, path following, collision avoidance and COLREGs, as well as an introduction to deep reinforcement learning. A short account of previous work essential to the framework utilised is also given. Next, the design and implementation details for the simulation environment, DRL algorithms and incorporation of the COLREGs can be found in Chapter 3. Chapter 4 outlines the methods employed for performance evaluation, as well as the training and results yielded by both the qualitative and the risk-based approach. Finally, Chapter 5 concludes and provides reflections on the thesis, in addition to suggestions for future work.

Chapter 2

Background

In this chapter, the necessary theoretical background is provided. In Section 2.1, the rationale behind the marine vessel model is presented. Theory on path following and collision avoidance, including the relevant COLREGs and measures of collision risk, can be found in Section 2.2.1. Further, Section 2.3 provides the reader with sufficient theory on deep reinforcement learning, before ending on a short account of feasibility pooling – an algorithm developed by previous students and used in this thesis.

2.1 Modelling of marine vessels

In order to develop a DRL-based system for path following and collision avoidance, it is necessary for the agent to interact with an environment, so that it can learn. Hence, such a model needs to be in place even though the agent is not provided with a model of the environment directly. A key element of the environment is the dynamics of the vessel itself. Therefore, an introduction to the theory used as a basis for the simulation environment is presented here, which is based on *Handbook of Marine Craft Hydrodynamics and Motion Control* by T. I. Fossen [48].

Twelve equations of motion accurately describe the kinematics and kinetics of a marine vessel. Six of these represent the *kinematics* of the vessel, which are the relations between positions and velocities. The remaining six represent the dynamics, or *kinetics*, of the vessel – the relations between forces, moments, and the momentum. Before expressing the equations of motion, the framework in which

they are expressed must be introduced.

A marine vessel operates in three-dimensional space with 6 degrees of freedom (DOF), meaning that its configuration is described by six coordinates. Three of these represent the position of the ship, and are called *surge*, *sway*, and *heave*. The remaining three represent the orientation of the ship, called *roll*, *pitch*, and *yaw*. For an overview of the notation commonly used in conjunction with the six coordinates, see Table 2.1.1.

Table 2.1.1: Nomenclature for motion of marine vessels [48], as defined by SNAME [49].

	DOF	Forces/moments	Linear/angular velocities	Positions/Euler angles
surge	1	X	u	x
sway	2	Y	v	y
heave	3	Z	w	z
roll	4	K	p	ϕ
pitch	5	M	q	θ
yaw	6	N	r	ψ

Assumption 1. *(Local navigation.) The craft navigates at a local scale.*

The positions, velocities, forces, and moments are expressed relative to a frame of reference. To accurately describe the motion of a marine vessel moving along the surface of the Earth, a coordinate frame located in Earth's center of gravity should be used as reference. As a result, it is usually considered an inertial coordinate frame, $\{i\} = (x_i, y_i, z_i)$. However, since many ships do not navigate globally, but rather locally, the North-East-Down (NED) frame, denoted $\{n\} = (x_n, y_n, z_n)$, is often used for simplicity. The X-Y plane of the NED frame is defined as the tangential plane on the surface of the Earth, with the origin moving with the vessel. The NED frame can therefore be seen as the linear position of the vessel relative to the inertial frame. Further, the coordinate frame aligned with the position and orientation of the vessel is called the body frame, $\{b\} = (x_b, y_b, z_b)$. Specifically, the x-axis is aligned with the aft-to-fore axis, the y-axis points starboard, and the z-axis points from the top to bottom of the vessel. The position of the vessel is best described in the NED frame, whilst velocities, forces, and moments are better described in the body frame. Using vector notation, such that for instance $\boldsymbol{\omega}_{nb}^{i}$ denotes angular velocity of $\{b\}$ with respect to $\{n\}$ expressed in $\{i\}$, the positions, velocities, forces and moments can be denoted:

NED position	$\boldsymbol{p}_{nb}^{n} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3$	Attitude	$\boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathcal{S}^3$
Body-fixed linear velocity	$\boldsymbol{v}_{nb}^{b} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3$	Body-fixed angular velocity	$\boldsymbol{\omega}_{nb}^{b} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3$
Body-fixed force	$\boldsymbol{f}_{b}^{b} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \in \mathbb{R}^3$	Body-fixed moment	$\boldsymbol{m}_{b}^{b} = \begin{bmatrix} K \\ M \\ N \end{bmatrix} \in \mathbb{R}^3$

Here, $\mathbb{R}^3$ is the three-dimensional Euclidean space, and $\mathcal{S}^3$ is the sphere defined by three angles on the interval $[0, 2\pi)$. To simplify the equations of motion, the vectors are grouped together so as to form a pose vector $\boldsymbol{\eta}$, velocity vector $\boldsymbol{\nu}$ and force vector $\boldsymbol{\tau}$, as in Eq. 2.1.1.

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}_{nb}^{n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} \in \mathbb{R}^3 \times \mathcal{S}^3, \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}_{nb}^{n} \\ \boldsymbol{\omega}_{nb}^{b} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \in \mathbb{R}^6, \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{f}_{b}^{b} \\ \boldsymbol{m}_{nb} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix} \in \mathbb{R}^6 \tag{2.1.1}$$

Next, the assumption of calm sea is made both for simplification and to allow the focus to remain on COLREG-compliance.

Assumption 2. *(Calm sea.) No external forces, such as wind, waves, and ocean currents, act on the craft.*

Neglecting wind and current and assuming local navigation (and thus no hydrostatic forces), the equations of motion for a marine vessel can be expressed as:

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu}) + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} &= \boldsymbol{\tau} \end{aligned} \tag{2.1.2}$$

Here, $\mathbf{M}$ is the rigid-body mass matrix, $\mathbf{C}(\boldsymbol{\eta})$ is the rigid-body Coriolis and centripetal matrix due to rotation of the body frame about the inertial frame, and $\mathbf{D}(\boldsymbol{\eta})$ is the vessel's damping matrix. $\boldsymbol{J}(\boldsymbol{\eta})$ is the transformation matrix from the body to the NED frame, as given by

$$\boldsymbol{J}(\boldsymbol{\eta}) = \begin{bmatrix} \mathbf{R}(\boldsymbol{\Theta}_{nb}) & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{T}(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \tag{2.1.3}$$

$\mathbf{R}(\boldsymbol{\Theta}_{nb})$ is the linear velocity rotation matrix, and $\mathbf{T}(\boldsymbol{\Theta}_{nb})$ is the angular velocity transformation. Writing sin, cos, and tan as s, c, and t, the rotation matrix and transformation are given as in Eq. 2.1.4 and Eq. 2.1.5. Here, a rotation sequence ϕ-θ-ψ was chosen as an example.

$$\mathbf{R}_b^n(\boldsymbol{\Theta}_{nb}) = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} = \begin{bmatrix} \mathrm{c}\psi & -\mathrm{s}\psi & 0 \\ \mathrm{s}\psi & \mathrm{c}\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{c}\theta & 0 & \mathrm{s}\theta \\ 0 & 1 & 0 \\ -\mathrm{s}\theta & 0 & \mathrm{c}\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ 0 & \mathrm{s}\phi & \mathrm{c}\phi \end{bmatrix} \tag{2.1.4}$$

$$\mathbf{T}(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & \mathrm{s}\psi\mathrm{t}\theta & \mathrm{c}\phi\mathrm{t}\theta \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ 0 & \mathrm{s}\phi/\mathrm{c}\theta & \mathrm{c}\phi/\mathrm{c}\theta \end{bmatrix} \tag{2.1.5}$$

Assumption 3. *(Horizontal-plane model.) The craft only moves in the horizontal plane, with no fluctuations in heave, roll, or pitch.*

Since this work deals with navigation in the horizontal plane with environmental forces neglected, it is natural to reduce the model to a 3-DOF model for simplicity. Doing so gives generalised coordinates $\boldsymbol{\eta} = [x^n, y^n, \psi^n]^\top$, with velocity vector $\boldsymbol{\nu} = [u, v, r]^\top$. The 3-DOF model thus becomes

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \mathbf{R}_{z,\psi}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu}) + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} &= \boldsymbol{\tau} \end{aligned} \tag{2.1.6}$$

For path following in the horizontal plane, usually accomplished through course control, four variables called *heading*, *course*, *crab*, and *sideslip* are especially

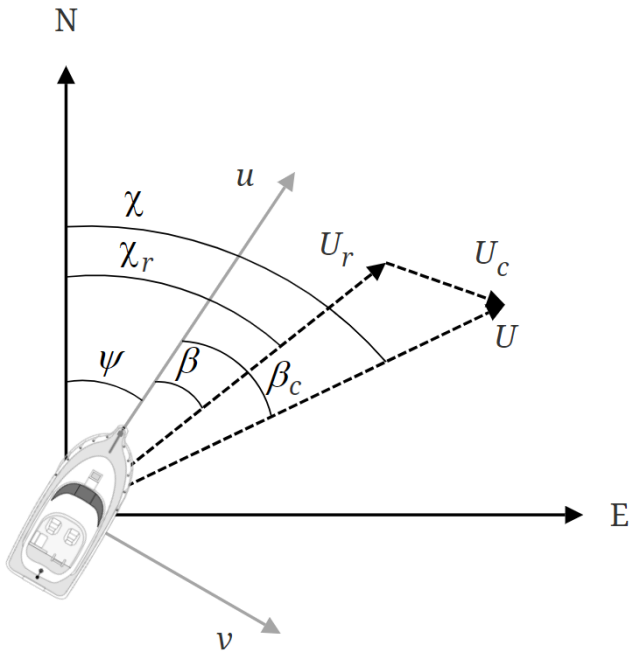


Figure 2.1.1: Ocean current triangle for the horizontal plane.

important. The heading ψ is defined as the angle between the aft-to-fore axis of the vessel and the true north. Defining $U = \sqrt{u^2 + v^2}$ as the horizontal-plane speed of the vessel, the course χ is defined as the angle between U and the true north, whilst the crab angle β_c is defined as the angle between U and the aft-to-fore axis of the vessel. Lastly, the sideslip angle is defined as the angle between the aft-to-fore axis of the vessel and the *relative* speed, $U_r = \sqrt{(u - u_c^2) + (v - v_c)^2}$, where u_c and v_c are the horizontal components of the velocity of the current. A visualisation of these angles is shown in Fig. 2.1.1.

The course angle can also be expressed as the sum of the heading and crab angles, which is an important relationship often used in course control:

$$\chi = \psi + \beta_c \tag{2.1.7}$$

However, when there is no wind or current present, as is assumed here, the sideslip and crab angles are equal, giving $U_r = U$.

2.2 Path following and collision avoidance for marine vessels

Here, a short introduction to classical path following and collision avoidance is given for comparative purposes. Path following and collision avoidance are closely connected and can be seen as competing objectives, as collision avoidance is essentially a deviation from a preassigned trajectory in the face of obstacles. In Section 2.2.1, the basics of a common path following algorithm for marine vessels is presented, and in Section 2.2.2, a short definition of collision avoidance and the type of system aimed for in this work are given. It should be noted that the sections on path following are based on *Handbook of Marine Craft Hydrodynamics and Motion Control* by T. I. Fossen [48].

2.2.1 Path following

In path following, the goal is for a vessel to converge to a predefined reference path. Since the path given is parameterised without the use of time, the controller only responds to spatial variables, such as Euclidean distance to the path. In *trajectory* tracking, importance is also given the temporal error.

A common method for straight-line path following is line-of-sight (LOS), where the path is represented using waypoints expressed in the NED frame. Defining a kth waypoint as $\boldsymbol{p}_k^n = [x_k, y_k]^\top$ for $k = 1, ..., n$, the path is comprised of the straight-line segments between subsequent waypoints $\boldsymbol{p}_k^n$ and $\boldsymbol{p}_{k+1}^n$. A positive angle α_p can then be found between a straight-line segment and the x-axis of the NED frame, called the *path tangential angle*:

$$\alpha_p = \text{atan2}(y_{k+1} - y_k, x_{k+1} - x_k) \tag{2.2.1}$$

The function `atan2` is used to ensure the calculation of the correct angle by taking the sign of the x-coordinates into account. To get the positive angle, the result belonging to the interval $[-\pi, \pi]$ must be mapped to $[0, 2\pi]$. Doing so, the position $\boldsymbol{p}^n = [x, y]$ of the vessel can be expressed in a path-fixed coordinate frame by rotating by α_p around the z^n-axis and translating the origin to $\boldsymbol{p}_k^n$. The corresponding rotation matrix is given by

$$\boldsymbol{R}(\alpha_p) = \begin{bmatrix} \cos\alpha_p & -\sin\alpha_p \\ \sin\alpha_p & \cos\alpha_p \end{bmatrix} \tag{2.2.2}$$

Combining rotation and translation, the path-fixed coordinates can be found (Eq. 2.2.3). These coordinates are often referred to as the *along-track error*, x_e, and the *cross-track error* (CTE), y_e.

$$\boldsymbol{p}_e = \begin{bmatrix} x_e \\ y_e \end{bmatrix} = \boldsymbol{R}(\alpha_p)^\top (\boldsymbol{p}^n - \boldsymbol{p}_k^n) \tag{2.2.3}$$

The purpose of transforming the vessel position to the path-fixed frame is to align the x-axis of the reference frame with the path. In this way, the control objective is simply to diminish the CTE, such that:

$$\lim_{t \to \infty} y_e(t) = 0 \tag{2.2.4}$$

To achieve smooth transitions between the waypoints, an acceptance region R_k around a waypoint $\boldsymbol{p}_k^n$ is defined. Doing so, the autopilot looks to the next waypoint once the ship has entered the acceptance region of the current waypoint. Circular acceptance regions are most commonly used, and a widely used heuristic for the acceptance region is $2L_{pp}$, where L_{pp} is the ship length. The switching condition for the autopilot is thus:

$$(x_k - x)^2 + (y_k - y)^2 \leq R_k^2 \tag{2.2.5}$$

In classical control, the vessel course can then be controlled using for instance a PID controller.

In this work, however, a smooth path is constructed as a parametric curve, complicating the calculation of the CTE slightly. Choosing the arc-length ω as the function parameter, the path can be parametrised as

$$\boldsymbol{p}_d(\omega) = [x_d(\omega), y_d(\omega)]^\top \tag{2.2.6}$$

where $x_d(\omega)$ and $y_d(\omega)$ are given in the NED-frame. Since the path is defined as a function of ω, we can find the value of ω corresponding to the closest point on the path, denoted $\bar{\omega}$, via the optimisation problem

$$\bar{\omega} = \arg\min_{\omega} \left(x^n - x_d(\omega)\right)^2 + \left(y^n - y_d(\omega)\right)^2 \tag{2.2.7}$$

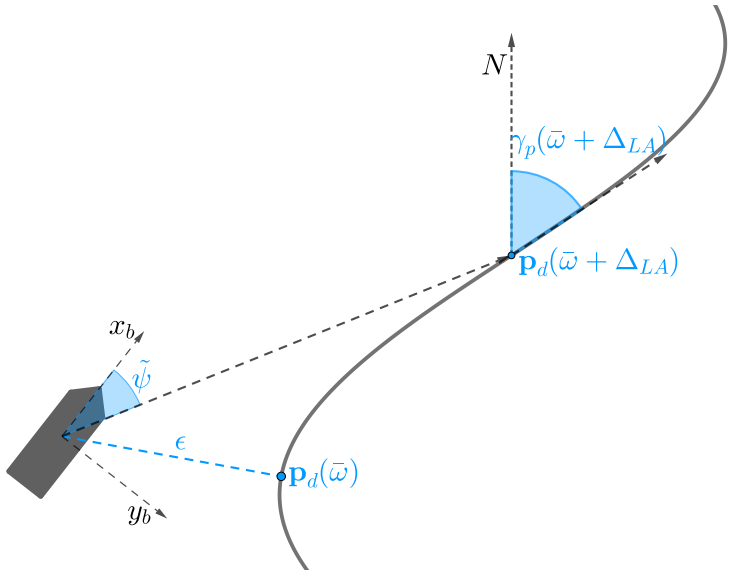


Figure 2.2.1: Illustration of key concepts for navigation with respect to path following. The path reference point $\boldsymbol{p}_d(\omega)$, i.e. point yielding the closest Euclidean distance to the vessel, is here located right of the vessel, while the look-ahead reference point $\boldsymbol{p}_d(\bar{\omega} + \Delta_{LA})$ is located a distance Δ_{LA} further along the path [5].

In turn, the CTE, illustrated in Fig. 2.2.1, can easily be found by first calculating the point on the path corresponding to the path variable $\bar{\omega}$, before applying it to the equation below:

$$y_e = \left\| [x^n, y^n]^T - \boldsymbol{p}_d(\bar{\omega}) \right\| \tag{2.2.8}$$

To provide the RL agent with information about the path ahead, allowing for smoother and more sensical behaviour, it is useful to consider the look-ahead distance Δ_{LA}. Setting the desired course angle equal to the direction to the corresponding point on the path, $\boldsymbol{p}_d(\bar{\omega} + \Delta_{LA})$, the look-ahead distance can be used to adjust the trade-off between path following and actuation. Following this scheme, the heading error $\tilde{\psi}$ is defined as the deviation between the vessel heading ψ and the direction towards the look-ahead point, illustrated in Fig. 2.2.1. The heading error is calculated according to

$$\tilde{\psi} = \text{atan2}\left(\frac{y_d(\bar{\omega} + \Delta_{LA}) - y^n}{x_d(\bar{\omega} + \Delta_{LA}) - x^n}\right) - \psi \tag{2.2.9}$$

In an attempt to further improve the agent's ability to maneuver in a smooth manner, the look-ahead heading error $\tilde{\psi}_{LA}$ is introduced. Defining the path angle α_p as the angle between the derivatives x'_p and y'_p at the point corresponding to the arc-length $\bar{\omega}$ gives

$$\alpha_p(\bar{\omega}) = \text{atan2}(y'_p(\bar{\omega}), x'_p(\bar{\omega})) \tag{2.2.10}$$

Finally, the look-ahead heading error is then given as the difference between the path angle at the look-ahead point, $\alpha_p(\bar{\omega} + \Delta_{LA})$, and the heading of the OS, ψ:

$$\tilde{\psi}_{LA} = \alpha_p(\bar{\omega} + \Delta_{LA}) - \psi \tag{2.2.11}$$

2.2.2 Collision avoidance

It is challenging to find a clear-cut definition of collision avoidance systems. In essence, it is a system which *aims to enhance the active safety of the vehicle* [50]. This can be achieved through advanced breaking controllers, as proposed in the article cited, or by evading the obstacle through course controllers.

A collision avoidance system for a marine vessel can be structured in many ways, but is generally focused on changing the vessel's course according to the COLREGs. Several of the most important algorithms used for such systems today are discussed in the literature review in Section 1.2. Moreover, as touched upon in the review, a marine collision avoidance system is normally either comprised of a single unit or a set of modules arranged in cascade. In a modular system, the higher level modules are deliberative, and deal with long-term planning. Low-level modules are reactive, responding rapidly to sudden changes in the environment.

Although an RL approach allows either an end-to-end system or a cascaded architecture, the former is employed in this work. This means that the system takes an observation vector containing information about the vessel state and environment as input, and outputs the desired control input directly.

In either case, the collision avoidance system makes part of the *control system* of a vessel, which in turn is one of the modules in a guidance, navigation and control (GNC) system, illustrated in Fig. 2.2.2. Due to the difference in computational

load, the high-level guidance system runs at a lower frequency than the low-level motion control system – enabled by the decoupling of the systems.

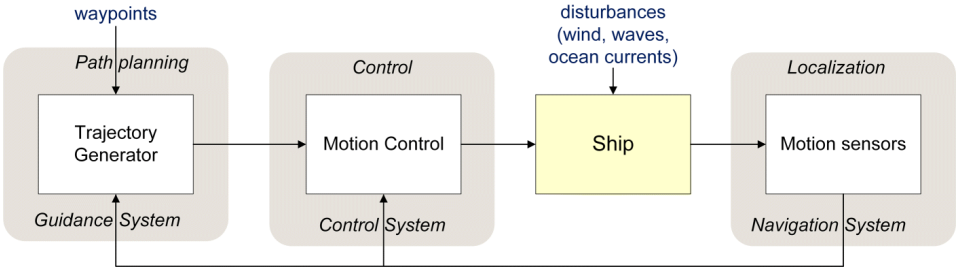


Figure 2.2.2: GNC system architecture [51]

2.2.3 Relevant COLREG rules

Below, the relevant sections of the International Regulations for Preventing Collisions at Sea [52] are presented. As can be seen, the two main takeaways from these rules are that 1) the give way vessel should take early and substantial action, and 2) safe speed should be ensured at all times, such that course alteration is effective towards avoiding collisions where there is sufficient sea-room. Since rules 6 and 8 are especially difficult to quantify, compliance to rules 14-16 is the focus of the work.

Rule 6: Safe speed

> *Every vessel shall at all times proceed at a safe speed so that she can take proper and effective action to avoid collision and be stopped within a distance appropriate to the prevailing circumstances and conditions.*

Rule 8: Action to avoid collision

> *(b) Any alteration of course and/or speed to avoid collision shall, if the circumstances of the case admit, be large enough to be readily apparent to another vessel observing visually or by radar; a succession of small alterations of course and/or speed should be avoided.*
>
> *(c) If there is sufficient sea-room, alteration of course alone may be the most effective action to avoid a close-quarters situation provided that it is made in good time, is substantial and does not result in another close-quarters situation.*
>
> *(d) Action taken to avoid collision with another vessel shall be such as*

to result in passing at a safe distance. The effectiveness of the action shall be carefully checked until the other vessel is finally past and clear.

(e) If necessary to avoid collision or allow more time to assess the situation, a vessel shall slacken her speed or take all way off by stopping or reversing her means of propulsion.

Rule 14: Head-on situation

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

(b) Such a situation shall be deemed to exist when a vessel sees the other ahead or nearly ahead and by night she could see the masthead lights of the other in a line or nearly in a line and/or both sidelights and by day she observes the corresponding aspect of the other vessel.

(c) When a vessel is in any doubt as to whether such a situation exists she shall assume that it does exist and act accordingly.

Rule 15: Crossing situation

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

Rule 16: Action by give-way vessel

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

2.2.4 Measures of collision risk

The rules presented in Section 2.2.3 were intended for human interpretation, and contain ambiguities such as "large enough" (Rule 8) and "substantial action" (Rule 16). How can they be translated into a form that can be used in reinforcement learning? An important first step is recognising the relationship between the COLREGs and collision risk. The COLREGs are in place to reduce collision risk, but also affect the risk level indirectly by influencing the probable behaviour of the target ship (TS). Since there is such an interdependence between the rules and the

risk level, employing a measure of risk as a proxy for the COLREGs may prove useful.

Analysing the historical trends of measuring collision risk, three main developments in methods used can be observed [14]:

1. Traffic flow theory
2. Ship safety domains
3. Collision risk indexes

The initial efforts to quantify collision risk were based on *traffic flow theory*, a method built on empirical studies and statistical analysis of traffic in specific water areas. For instance, Cookcroft [53] investigated the collision rates for ships of varying tonnage relative to their position in a water area. Goodwin [54] took it further and studied the rate of dangerous encounters.

As statistical analysis of historical data was deemed insufficient for dynamic collision avoidance, *ship safety domains* were introduced. The ship safety domain defines a region around the ship in question which should not be entered by other ships. Hence, there is a risk of collision if one ship is inside the safety domain of another, and the ship domain can be said to be a generalisation of a safe distance [55]. When applying the ship domain to an encounter situation in order to determine risk, one of the four *safety criteria* are normally used:

1. the OS domain should not be violated by a TS
2. a TS domain should not be violated by the OS
3. neither of the ship domains should be violated
4. ship domains should not overlap, such that they remain mutually exclusive.

In the more recent contributions of Rawson et al. [56] and Wang and Chin [57] in 2014 and 2016, the last criterion of non-overlapping ship domains is used.

It is important to note that a ship domain is usually defined depending on which situation the ship finds itself in, in order to respect the COLREGs. For instance, the domain used while the OS is overtaking another ship is symmetrical with origin coinciding with the centre of the OS. In a head-on situation, on the other hand, the origin is shifted to the right of the OS, as close encounters on the starboard side are to be avoided.

Davis et al. [58] expanded the theory of ship safety domains in their well-known work on *ship arenas*. The ship arena defines the distances around the OS at which

action should be made to avoid a dangerous encounter, and is therefore larger than the ship safety domains proposed initially. In addition to the OS's length and velocity, the distance to the closest point of approach (DCPA) and the time to the closest point of approach (TCPA) are used to construct the limits of the ship arena. A geometrical representation of DCPA and TCPA are presented in Fig. 2.2.3, giving rise to the equations

$$DCPA = R\sin(\chi_R - \chi_{OS} - \theta_T - \pi) \tag{2.2.12}$$

and

$$TCPA = \frac{R}{V_R}\cos(\chi_R - \chi_{OS} - \theta_T - \pi) \tag{2.2.13}$$

where R is the absolute distance between the OS and TS, and V_R and χ_R are the relative speed and course between them. In addition, χ_{OS} is the course of the OS, while θ_T is the bearing of the TS relative to the OS.

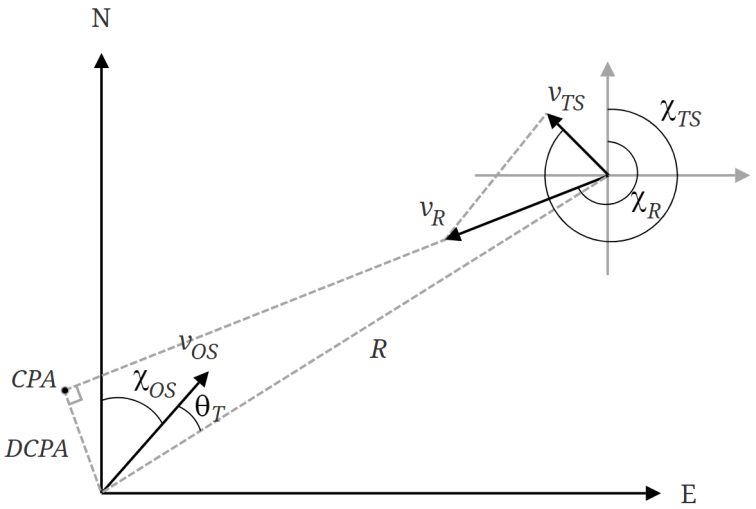


Figure 2.2.3: Geometric representation of DCPA and DCPA.

This leads to the next development in collision risk evaluation, namely *collision risk indexes* (CRIs), which are based largely on the DCPA and TCPA. In addition, a CRI can include the absolute distance from the OS to the TS R, velocity ratio K of two encountering ships, relative course χ_R, and other key features. Recently,

simple CRIs alone are considered unable to capture the gradual and complex nature of collision risk. As a result, it has become the norm to combine the CRI with fuzzy logic or the fuzzy comprehensive evaluation method. In fuzzy logic, fuzzy IF-THEN rules are applied to the parameters involved, such as DCPA and TCPA, in order to determine the level of risk. In the fuzzy comprehensive evaluation method, on the other hand, *membership functions* $u(\cdot) \in [0, 1]$ are used instead of IF-THEN rules, taking more details into account. The final CRI is then given as the weighted sum of the membership function outputs, as exemplified below:

$$CRI = \alpha_{DCPA} \cdot u_{DCPA}(DCPA) + \alpha_{TCPA} \cdot u_{TCPA}(TCPA) + \alpha_R \cdot u_R(R) \quad (2.2.14a)$$

$$\alpha_{DCPA} + \alpha_{TCPA} + \alpha_R = 1 \quad (2.2.14b)$$

2.3 Deep reinforcement learning

Machine learning (ML) is the branch of artificial intelligence (AI) that has received most attention over the past few decades, and is often what is referred to when talking about AI. Creating machines that can learn has proven immensely useful in tasks such as image recognition and natural language processing, and ML is now widely used by companies such as Facebook and Google [59]. Possibly the most popular type of ML is that of deep learning (DL), which requires large amounts of labelled training data. Reinforcement learning (RL), on the other hand, attempts to solve problems in a more intuitive fashion, through trial and error. The combination of DL and RL forms deep reinforcement learning (DRL) – a hybrid system described in section 2.3.3. First, however, it is necessary to take a closer look at RL and DL separately.

2.3.1 Reinforcement learning

In reinforcement learning, a problem is posed as an agent interacting with its environment. Although a plethora of RL algorithms exist, they share core features. In essence, the interplay between the agent and the environment is described by Fig. 2.3.1. As can be seen, the flow of information is represented by states, rewards, and actions. The state S_t provides the agent with key information about its position in the environment at time t, whilst the reward R_t provides feedback, expressing the value associated with the previous action taken. The reward signal is used to update the agent's understanding of its environment, and must be designed before

training the agent, called *reward shaping*. Based on the current understanding and information about state, the agent decides what to do – which action A_t to take. When an action is executed, a new state S_{t+1} and reward R_{t+1} are yielded.

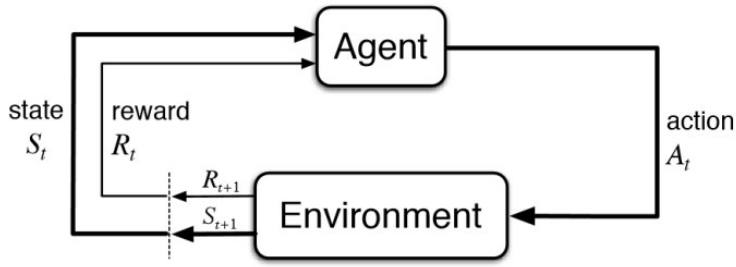


Figure 2.3.1: A representation of the interaction between the RL agent and environment [60].

It follows from the formulation of the RL agent and environment that the system does not depend on detailed a priori information, but rather learns about the consequences of actions through observations. As a result, RL allows for model-free solutions, meaning that the RL approach can be useful in situations where good models for the system at hand are unavailable, or where the model is complex. A potential benefit of learning is a more generalised mapping from the state-space to the action-space, which is a weakness in conventional approaches such as MPC.

2.3.1.1 Markov decision processes

A fundamental assumption in RL systems is that the problem to be solved is appropriately described as a Markov decision process (MDP). One of the key attributes of MDPs is that they have the Markov property. This means that the probability distributions of future states are dependent on the present state and action only. For MDPs, we can thus say that all the information about past states and actions needed to predict the future is baked into the present state and action, as shown in Eq. 2.3.1.

$$\begin{aligned}\mathbb{P}(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}, ...S_0 = s_0) \\ = \mathbb{P}(S_{t+1} = s' | S_t = s_t, A_t = a_t)\end{aligned} \tag{2.3.1}$$

In simple terms, an MDP is a system consisting of a set of possible states $s \in \mathcal{S}$ and a set of possible actions $a \in \mathcal{A}$, where each transition from one state to

another can be represented by a transition function and a reward function. The transition function $\mathcal{T}(s, a, s')$ models the transitions between states stochastically, and is equivalent to the probability that an action a takes the agent from a state s to a new state s'. The reward function $\mathcal{R}(s, a)$ describes the immediate reward given after completing this transition. In addition, an MDP is defined by an initial state s_0, or distribution, and possibly a terminal state s_f.

$$\mathcal{T}(s, a, s') = \mathbb{P}(S_{t+1} = s' | S_t = s_t, A_t = a_t) \tag{2.3.2}$$

$$\mathcal{R}(s, a) = \mathbb{E}(R_t | S_t = s_t, A_t = a_t) \tag{2.3.3}$$

The MDP framework was generalised by the introduction of partially observable Markov decision processes (POMDP), where the true state of a system is unobservable by the agent. As a result, an agent dealing with a POMDP assumes MDP dynamics, but must build beliefs about its environment through observation [61]. This can be done directly by constructing an internal model, or indirectly by iteratively registering the effects of actions over time. An illustrative example of an indirect way of solving a problem is rolling a die many times and continuously updating the mean, instead of calculating it directly using a known probability distribution for the outcomes.

2.3.1.2 Policies, value functions, and the Bellman equation

An RL problem can be modelled as a POMDP – or an MDP where the transition and reward functions are unknown. If they are known, the problem can be solved using classical optimisation approaches, such as value iteration or policy iteration. However, these approaches easily become computationally infeasible due to large state and action spaces. For stochastic processes, where a decision gives rise to a distribution of outcomes, this is especially true [62]. In order for an RL agent to learn which action to take given a specific state, it must have an understanding of the quality of an action, given this state, relative to a predefined criterion. Therefore, the first necessary component when building the RL framework is the *return*, denoted G, which is a cumulative reward resulting from said quality of action. Since the RL agent is always operating relative to predefined measures of quality, the return can be defined in any way that is conducive to solving the problem. For instance, the return can be defined as the sum of rewards over a finite horizon T. In this case, we place the same value on all rewards obtained over a

fixed window of time:

$$G_{finite} = \mathbb{E}\left[\sum_{k=t}^{t+T} R_k\right] \tag{2.3.4}$$

However, since a common goal is to solve a problem quickly and efficiently, the return is often formulated as the infinite-horizon discounted sum of rewards. This means that a higher value is placed on rewards that come sooner. Another reason for using a discounted return is that rewards that lie far into the future come with higher uncertainty. The discounted sum of rewards then incentivises the agent to base its decisions more on the rewards that lie closer in time, reflecting the unreliability of predictions about future states. Hence, we define a discount factor $\gamma \in [0, 1)$, such that the return becomes:

$$G_{discounted} = \mathbb{E}\left[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ... + \gamma^i R_{t+i}\right] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k}\right] \tag{2.3.5}$$

The next step is to introduce a few useful concepts with corresponding mathematical representations: the *policy*, the *state-value*, and the *action-value*. The policy is the mapping from state to action, or a sequence of decisions [62], and is essentially what the RL should learn. Since the process at hand is not necessarily deterministic, the policy function π can be either deterministic or stochastic. In most real applications, the policy is modelled as stochastic in order to handle uncertainty in the environment. A good policy can be achieved by learning from observations of rewards, and the *value functions* help learning by keeping track of these rewards. The state-value describes the expected utility of starting in a specific state s, given a policy π. The action-value is simply an extension of this, describing the expected utility of starting in a specific state s and taking a specific action a, given a policy π.

Mathematically, the state-value function v^π (Eq. 2.3.6) is the expected return, given a state s and a policy π. The state-value function is also commonly called the *value* function.

$$v^\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, \pi\right] \tag{2.3.6}$$

Due to the Markov property, we can define the state-value function as the expected sum of the immediate reward and the discounted value of the next state:

$$v^{\pi}(s) = \mathbb{E}\left[R_t + \gamma v^{\pi}(S_{t+1}) \middle| S_t = s, \pi\right] \tag{2.3.7}$$

Building on the state-value function, the action-value function q^{π} (Eq. 2.3.8) can be defined as the expected return. The action-value function is also called the *quality* function, and can be seen as a measure of the quality of a specific state-action pair.

$$q^{\pi}(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, A_t = a, \pi\right] \tag{2.3.8}$$

Using the same logic as before, the quality function can be defined as the expected sum of the immediate reward when taking action a, plus the discounted value of the next state:

$$q^{\pi}(s,a) = \mathbb{E}\left[R_t + \gamma v^{\pi}(S_{t+1}) | S_t = s, A_t = a, \pi\right] \tag{2.3.9}$$

The goal of the RL agent is to maximise the return, which is done by learning a policy that leads to maximal return. Since the optimal policy π^* is the policy that yields the optimal return, we define the optimal policy as in Eq. 2.3.10. Here, the optimal value function v^* is defined as the maximum action-value for state s, as given by the optimal action-value function q^*.

$$\pi^*(a|s) = \arg\max_{\pi} v^*(s) = \arg\max_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | \pi\right] \tag{2.3.10}$$

So, how do we solve the problem of finding the optimal policy? In the 1950s, E. Bellman introduced dynamic programming (DP), a method for solving a problem by breaking it down into sub-problems, and recursively solving those. To use DP, the problem must have optimal substructure, and exclusively consist of optimal sub-problems. MDPs satisfy both of these requirements. The value functions give us the framework for expressing the overlapping sub-problems, as the value can be found recursively. For the optimal substructure, we utilise a core principle of DP, namely the *principle of optimality*. In essence, the principle of optimality states that an optimal policy for a given state is independent of the previous states, meaning that the problem has optimal substructure. For MDPs, the Bellman

equation (Eq. 2.3.11), or principle of optimality equation, provides the necessary condition for fulfilling the principle of optimality.

$$v^*(s) = \max_{a \in \mathcal{A}} q^*(s, a) \tag{2.3.11}$$

Using Eq. 2.3.9 to substitute for q^* in Eq. 2.3.11, and vice versa, the recursive equations for solving the RL problem can be formulated:

$$\begin{aligned} v^*(s) &= \max_{a \in \mathcal{A}} q^*(s, a) \\ &= \max_{a \in \mathcal{A}} \mathbb{E}\left[R_t + \gamma v^*(S_{t+1}) | S_t = s, A_t = a, \pi^*\right] \\ &= \max_{a \in \mathcal{A}} \sum_{s', r} \mathcal{T}(s, a, s') \left[r + \gamma v^*(s')\right] \end{aligned} \tag{2.3.12}$$

$$\begin{aligned} q^*(s, a) &= \mathbb{E}\left[R_t + \gamma \max_{a' \in \mathcal{A}} q^*(S_{t+1}, a') | S_t = s, A_t = a\right] \\ &= \sum_{s', r} \mathcal{T}(s, a, s') \left[r + \gamma \max_{a' \in \mathcal{A}} q^*(s', a')\right] \end{aligned} \tag{2.3.13}$$

2.3.1.3 Learning methods

As mentioned in Section 2.3.1.2, the Bellman equation can be solved explicitly using a method such as value or policy iteration. However, these methods are computationally expensive, and very slow for problems with large state- and action-spaces. Through RL, the value and policy functions can be approximated, giving a computationally simpler problem. At the same time, the exploration and learning might take time.

There are many learning methods available within RL. One way to categorise the algorithms is by distinguishing between model-based and model-free methods. Model-based methods attempt to explicitly model the transition and reward functions, which can later be used for more long-term planning and optimisation. An example of a model-based algorithm is simulated policy learning (SimPLe) [63], which is based on video prediction models. Model-free methods, on the other hand, attempt to approximate the value functions by representing them as for instance tabular mappings from state-action pair to quality value. Important examples

of tabular model-free algorithms are Q-learning [64] and SARSA (State-Action-Reward-State-Action) [65]. Today, model-free algorithms are by far the most popular, although one might expect the introduction of more model-based algorithms in the future due to the extensive amount of training required for their model-free counterparts.

Another important distinction between RL algorithms is whether they are so-called *actor-critic*, *actor-only*, or *critic-only* algorithms. The policy of an RL agent is used to make decisions, and can thus be seen as the *actor* of the system. Conversely, the action-value function, or quality function, gives value to each state-action pair. It is therefore said to be the *critic*. In other words, actor-only methods approximate the policy function, critic-only methods approximate the action-value function, and actor-critic methods approximate both. Deep Q-learning (DQN) and SARSA are well-known critic-only methods, while actor-only methods have limited usability as they require a restricted policy space [66]. Actor-critic methods such as deep deterministic policy gradients (DDPG) [38] and proximal policy optimisation (PPO) [41] can be said to make use of the best of both worlds, and can be applied to continuous state and action spaces. PPO is described in greater detail in 2.3.3.3.

In order to approximate the value functions, the RL agent must learn through observations. As decent amounts of well-balanced real data is often difficult to obtain, letting the agent learn through simulations might be beneficial. When running simulations, it is common to define an *episode* as either a certain amount of single steps (actions) taken in the environment, or as the period up until a specific event, such as the loss of a life in a game. Updates to the value functions may be done *off-line* after an episode, or *on-line* after every single step or after a specific number of steps within an episode. The former is called *Monte Carlo* learning, and has the disadvantages of high variance and no exploitation of the Markov property [67]. *Temporal Difference* (TD) learning, on the other hand, exploits the Markov property and has relatively low variance [67]. Therefore, it is often used in RL applications. TD learning works by making adjustments to the state-value and action-value functions after observing the reward received over n steps. Using a learning rate α and $n = 1$ number of steps, and denoting the next action and state as a' and s', v and q can be approximated as follows:

$$V(s) = V(s) + \alpha[r + \gamma V(s') - V(s)] \tag{2.3.14}$$

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{2.3.15}$$

The above equations describe how the value functions are updated, but not how the next action, a', is chosen. SARSA is an *on-policy* algorithm, which means that the behaviour policy is the same as its update policy, as in Eq. 2.3.15. In SARSA, both the value of an action and the next action are chosen based on an ϵ-*greedy* algorithm that chooses a random action a certain percentage of the time and the action with the highest expected return the rest of the time. *Off-policy* algorithms such as Q-learning and PPO choose the next action independently of the current policy. For instance, Q-learning learns greedily, always maximising the expected action-value, but determines the next action in an ϵ-greedy fashion.

2.3.2 Deep learning

Deep learning (DL) is a subcategory of machine learning in which deep neural networks (DNNs) are used to approximate functions. This is traditionally done by training the network on a large amount of data, through supervised learning. In RL problems, the RL agent generates training data by taking note of the reward received as a result of a specific state-action pairing. The key idea behind this method is seeing the problem at hand as a black box, and iteratively adjusting the parameters of the model to decrease the error between the prediction and the correct *labels*. Due to the high level of customisation possible, DL can be applied to a wide range of problems where the function or model at hand is complex or unknown. The two main types of problems tackled by DL are regression and classification problems, which aim to approximate a mapping function from an input to a continuous and discrete output, respectively.

2.3.2.1 Artificial neural networks

Artificial neural networks (ANNs) consist of artificial neurons, or *nodes*, arranged in a network made up of *layers*, as illustrated in Fig. 2.3.2. A simple neural network is built up of three such layers: an *input* layer, a *hidden* layer, and an *output* layer. A DNN is an ANN with more than one hidden layer.

The architecture of an ANN is a function of the types of nodes present in the network, as well as how they are connected. In a feed-forward network, such as the one illustrated, the flow of information only travels in one direction through the network. In addition, no memory of past states is held. The layers are also *dense*, meaning that each node connects to all the nodes in the next layer. However, many types of ANNs with different functionalities exist, and can be implemented

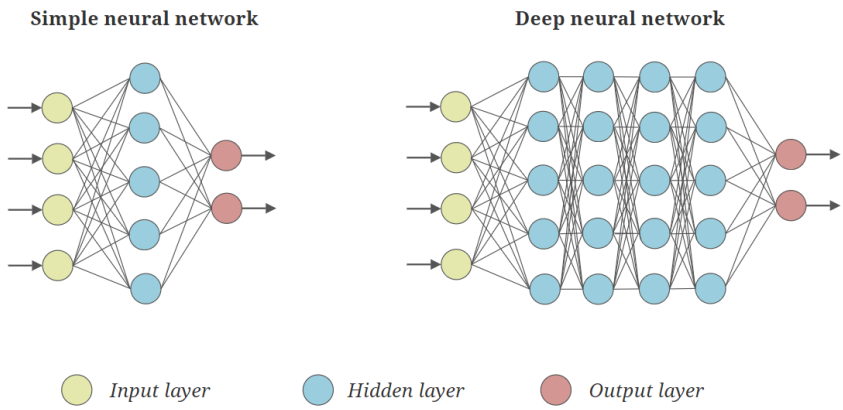


Figure 2.3.2: Simple and deep neural networks.

in conjunction with RL algorithms. For instance, recurrent NNs make predictions based on both present and past states, as they contain feedback loops. In long short-term memory (LSTM) networks, a type of recurrent network, a memory cell called an LSTM unit is implemented to handle the vanishing gradient problem that is prone to occur in RNNs. Another widely used type of layer is the *convolutional* layer, where filters are applied to the input data. Convolutional layers are especially useful when processing images or other data that contains higher level features that can be detected by filters.

Deep learning works by finding meaning in data through learning, which means that their only source of information is the input data. If this data is skewed, or the network is over-trained on a specific dataset, it might *overfit* on the data used. As a result, proper pre-processing of the data is necessary to avoid poor results. The most primitive way of reducing overfitting is *early stopping*, which is simply monitoring the performance of the network on a validation data set that has not been used for training. It is also common to use *batch learning*, where the network learns from a specific number of datapoints at once, reducing the sensitivity to individual datapoints. Several other techniques have also been developed to generalise the training, called *regularisation*. In *L1* and *L2* regularisation, a term proportional to the weights in the neural network is added to the cost function. The idea behind these techniques is that large weights indicate overfitting, as very high importance is given to a specific input or feature. *Dropout* is another widely used technique, where a percentage of weights are set to zero at each iteration. This way, dependency on specific inputs is prevented.

Convergence is not guaranteed for neural networks, but some techniques exist to improve the speed, performance, and stability. When creating an ANN, the weights must be initialised. Weight initialisation is often done using a normal distribution. However, other initialisation methods can work better for certain types of activation functions. For instance, *Xavier* and *He* initialisation were developed for the tanh and ReLU activation functions, respectively. In these initialisation methods, the number of input neurons is taken into account, such that the variance of the layer is reduced. By doing so, the network learns better, as the gradients do not explode or vanish as easily. *Batch normalisation* improves learning speed and stability during training by converting a the output of a layer to zero-mean and unit variance, before it acts as input to the next layer. In addition, it can be mentioned that batch learning increases stability by reducing the fluctuations that could arise from making updates to the network based on single data points.

2.3.2.2 Artificial neurons

The basic building block in a deep neural network is the artificial neuron. As the nomenclature reveals, these are inspired by the neurons found in biological nervous systems. Like their natural counterparts, artificial neurons produce an output based on its inputs and parameters. The simplest artificial neuron was introduced by F. Rosenblatt in the late 1950's, and is called a *perceptron*. A visualisation of the perceptron with its inputs x_i, weights w_i and output y is shown in Fig. 2.3.3.

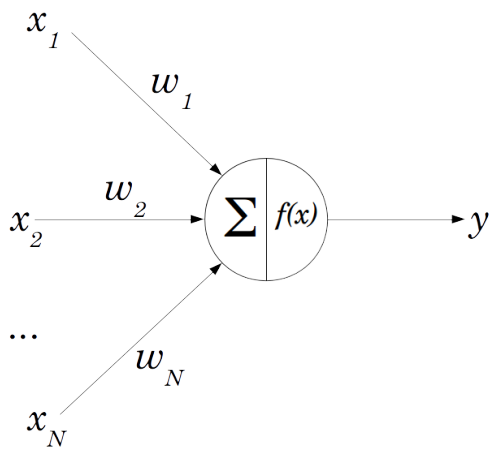


Figure 2.3.3: The perceptron [68]

If the sum of the weighted inputs is higher than a specified threshold, the perceptron

is excited. If the weighted sum is below the threshold, the output is zero (Eq. 2.3.16). In essence, the perceptron makes decisions or predictions based on the input data.

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j < \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases} \tag{2.3.16}$$

One downside when using perceptrons is that a small change in the input can cause a drastic change in the output. In addition, only linear relationships can be described. By using a nonlinear and smoother *activation function*, training becomes easier and more complex correlations between inputs and outputs can be modelled. The S-shaped sigmoid and hyperbolic tangent functions have been widely used for this purpose, but suffer from slow convergence and the vanishing gradient problem (described in Section 2.3.2.3) due to saturation. As a result, the rectified linear unit (ReLU) has become increasingly popular. ReLU is simply $y = \max(\sum_j w_j x_j, 0)$, meaning that it is zero for negative weighted sums, and equal to the weighted sum for positive sums. Some other versions of the ReLU, such as leaky ReLU with a slight negative gradient for sums below zero, have also been introduced to further tackle the problem of vanishing gradient that might arise due to saturation for negative values. For classification problems, the softmax function is often used for the output layer. This way, the sum of the outputs in the output layer is unity, which means that the outputs can be seen as probabilities.

2.3.2.3 Backpropagation and gradient descent

How do deep neural networks learn? The first step in learning from labelled data is passing the data through the network, a so-called *forward pass*. Then, a *loss function*, or objective function, is used to calculate the loss. Mean square error (MSE) is a metric commonly used as loss (Eq. 2.3.17). Here, the y_{pred} is the ground truth value, and y_{target} is the output of the network, or the predicted value. n is the *batch size* – the number of data points passed through the network between each update to the parameters.

$$MSE = \frac{1}{2n}(y_{target} - y_{pred})^2 \tag{2.3.17}$$

The calculated loss can then be used to adjust the parameters of the network, which is called *backpropagation*. To minimise the error, an optimisation algorithm called *gradient descent* is used when updating the weights and biases. When the

batch size is set to 1, *stochastic* gradient descent is used, where the update rule is given by Eq. 2.3.18. E corresponds to the loss, α is the learning rate, and w_{ij} is the weight given to the output from neuron i in one layer to neuron j in the subsequent layer. Other methods are *batch* gradient descent for larger batch sizes, and *mini-batch* gradient descent, which takes the middle road.

$$w_{ij} = w_{ij} - \alpha \frac{\partial E}{\partial w_{ij}} \tag{2.3.18}$$

By calculating the partial derivatives of the error to the parameters of the last layer in the network, an adjustment can be made to each of these parameters in a way that corresponds to their influence on the error. In feed-forward networks, where the data is processed sequentially during the forward pass, the chain rule can be used to iteratively calculate the partial derivatives of the error to all the parameters in the network.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}} \tag{2.3.19}$$

$$= \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_j} y_i \tag{2.3.20}$$

$$= \frac{\partial E}{\partial y_j} \sigma_j'(z_j) y_i \tag{2.3.21}$$

Here, $z_j = \sum_{k=1}^{n} w_{kj} y_k + b_j$ is the input to neuron j, and $\sigma_j'(\cdot)$ is the activation function of neuron j. The activation of neuron j and output from neuron i are given from the forward pass, and the partial derivative of E with respect to the output of neuron j can be calculated using results from previous iterations. For the output layer, this derivative is equal to the derivative of the loss function with respect to the output y_{pred}. By introducing notation for layers, the derivative of the error with respect to the output of the i^{th} neuron in the $(l-1)^{th}$ layer becomes

$$\frac{\partial E}{\partial y_i^{l-1}} = \sum_{j \in N_l} \left(\frac{\partial E}{\partial y_j^l} \frac{\partial y_j^l}{\partial z_j^l} \omega_{ij}^l \right) \tag{2.3.22}$$

where N_j is the number of neurons in layer l.

2.3.3 Deep reinforcement learning

Although a reinforcement learning framework does not immediately require deep neural networks, the combination of reinforcement learning and deep learning has become increasingly popular. In reinforcement learning, the ultimate goal is to find an appropriate behaviour policy. For problems exhibiting continuous state and action spaces, this is normally done by approximating the action-value and policy functions via an actor-critic method. In this work, the proximal policy optimisation algorithm is used, which is a policy gradient method based on trust region policy optimisation (TRPO). Before introducing the formal basis of PPO in Section 2.3.3.3, the basics of policy gradient methods and trust region methods are therefore covered in Sections 2.3.3.1 and 2.3.3.2. It should be noted that the following sections are based on [69], [41] and [70].

2.3.3.1 Policy gradient methods

As previously discussed, the main objective in reinforcement learning is to discover a stochastic behaviour policy π_θ that maximises the return. This policy is parametrised by for instance the weights and biases of a neural network, denoted θ. The optimal policy is therefore the policy with the optimal parameters, as defined by Eq. 2.3.23.

$$\theta^* = \arg\max_\theta \mathbb{E}\left[\sum_t r(s_t, a_t)\right] \tag{2.3.23}$$

Denoting the trajectory as τ and the return (sum of discounted rewards) as $R(\tau)$, the objective function $J(\theta)$ can be written as

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[R(\tau)\right] \tag{2.3.24}$$

Further, utilising the relations

$$\mathbb{E}_{x\sim p(x)}\left[f(x)\right] = \int p(x)f(x)dx \tag{2.3.25}$$

$$\nabla_\theta f(x) = f(x)\,\nabla_\theta \log f(x) \tag{2.3.26}$$

the objective function and its gradient can be expressed as in Eq. 2.3.27 and 2.3.28.

$$J(\theta) = \int \pi_\theta(\tau) R(\tau) d\tau \tag{2.3.27}$$

$$\begin{aligned} \nabla_\theta J(\theta) &= \int \nabla_\theta \pi_\theta(\tau) R(\tau) d\tau && (2.3.28) \\ &= \int \pi_\theta(\tau)\, \nabla_\theta \log \pi_\theta(\tau) R(\tau) d\tau && (2.3.29) \\ &= \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) R(\tau) \right] && (2.3.30) \end{aligned}$$

Since the gradient can be represented as an expectation, it can be estimated using samples, such that the policy gradient and update rule become

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \right) \left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \tag{2.3.31}$$

and

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{2.3.32}$$

The first term in Eq. 2.3.31 measures the likelihood of a trajectory under the current policy. By multiplying this with the rewards, maximising $J(\theta)$ leads to increased likelihood for trajectories with higher rewards. In addition, the change from multiplication to a sum of logarithmic terms counteracts the problems of vanishing and exploding gradients that often arise in deep learning. Further, N is the batch size used for optimisation, and T is the number of steps in an episode. In eq. 2.3.32, α is the learning rate of the system.

A problem with the policy gradient method above is its high variance and resulting issues with convergence. Naturally, one way to remedy this is to limit the variance of the gradient. Recognising the sum of the rewards as the action-value function $Q(s, a)$, we define an *advantage function* $A(s, a)$, such that

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{2.3.33}$$

where $V(s)$ is the value function. The subscript t indicates the point in time. Since the action-value function gives the expected reward given a specific action a taken

in state s and the value function expressed the expected reward when taking the best action available in state s, the advantage function is effectively a measure of whether the action taken is better or worse than the current policy. In this case, the value function $V(s)$ is used as *baseline*, and the rewards are thus recalibrated according to the current policy – which reduces the variance of the gradient.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})(Q(s_{i,t}, a_{i,t}) - V(s_{i,t})) \tag{2.3.34}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A(a_{i,t}|s_{i,t}) \tag{2.3.35}$$

More generally, the estimated policy gradient (PG) can be written as

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right] \tag{2.3.36}$$

where $\hat{\mathbb{E}}_t[\cdot]$ denotes the estimated expectation at time t, computes as an empirical average over a finite batch of samples equal to the one represented in Eq. 2.3.34. Similarly, $\hat{A}_t$ is an estimate of the advantage of action a_t given state s_t, often computed according to Eq. 2.3.33. The estimator $\hat{g}$ can be found by differentiating the objective

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right] \tag{2.3.37}$$

2.3.3.2 Trust region methods

The main challenge when using vanilla policy gradient methods is that blindly making updates according to the gradient may be problematic. When little attention is paid to the shape of the objective function and what learning rate is appropriate, the algorithm is prone to make leaps that effectively ruin the training process. A common way to address this issue is via trust region methods – of which trust region policy optimisation (TRPO) [69] is the most famous. These methods make sure that the expected reward does not decrease when performing updates to the policy.

According to the chain rule, the objective used in vanilla policy gradient optimisation, Eq. 2.3.37, can be written as

$$L^{IS}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \tag{2.3.38}$$

This objective employs *importance sampling* (IS), which allows state-actions to be sampled from the old policy $\pi_{\theta_{old}}$, meaning that L^{IS} can be seen as a surrogate loss. Applying IS alleviates some of the issues connected to on-policy methods such as TRPO and PPO.

Furthermore, trust region methods assume that only the local approximation of our objective is accurate. TRPO handles this by expanding the optimisation problem to include a constraint on the Kullback–Leibler (KL) divergence between the old and new policies $\pi_{\theta_{old}}$ and π_θ. KL divergence, also called *relative entropy*, is a common way of measuring the difference between probability distributions such as behavioural policies in reinforcement learning. The resulting objective function becomes

$$L^{TRPO}(\theta) = \max_\theta \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \tag{2.3.39}$$

$$\text{s.t.} \quad \hat{\mathbb{E}}_t \left[\mathrm{KL} \left[\pi_{\theta_{old}}(\cdot|s_t),\, \pi_\theta(\cdot|s_t) \right] \right] \leq \delta \tag{2.3.40}$$

This way, a constraint is imposed on the change in the policy possible under a single update, according to the hyperparameter δ.

2.3.3.3 Proximal policy optimisation

The proximal policy optimisation (PPO) method was first introduced as an attempt to mitigate the main drawbacks of the other methods that were available. The creators of PPO at OpenAI found that Q-learning, vanilla policy gradient methods and trust region policy optimisation (TRPO) methods all suffered from lack of either scalability, sample efficiency, or robustness [41]. They noted that TRPO was generally the most sample efficient and robust approach, and sought to develop a more less complex method that would still offer these advantages.

PPO is based on the objective function used in vanilla policy gradient optimisation, and adds a modification in the spirit of the trust region approach. Let $r_t(\theta)$ be the probability ratio from $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}(a_t|s_t)}}$, such that an unchanged policy yields $r = 1$. By adding a clipping function to the objective, as in Eq. 2.3.41, a lower bound is imposed on the unclipped objective. The lower bound is dependent on

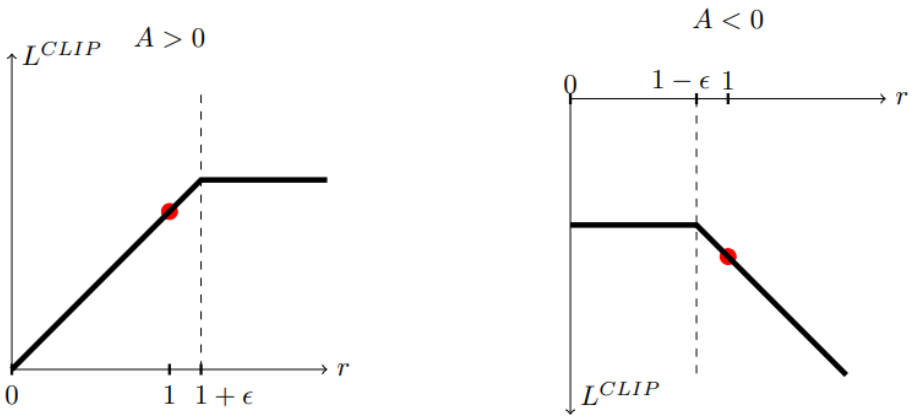


Figure 2.3.4: Plots showing a single timestep of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP} sums many of these terms [41].

the hyperparameter ϵ, which is often set to a value in the neighbourhood of 0.2. In practice, this means that large estimated improvements are ignored, which leads to a more stable training process overall.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] \tag{2.3.41}$$

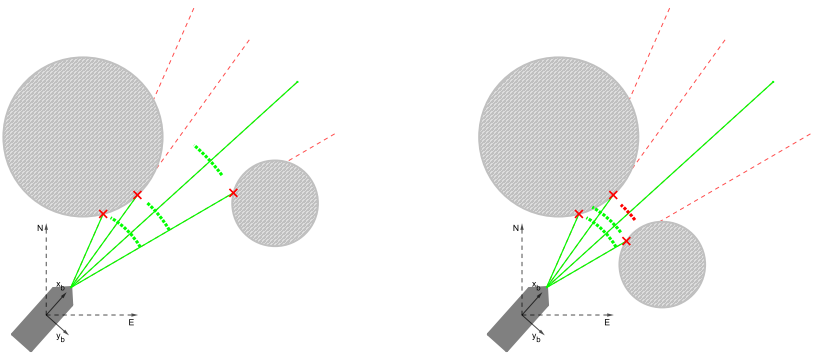
The objective function is represented graphically in Fig. 2.3.4.

2.4 Previous work

2.4.1 Feasibility pooling

One of the key features of the framework developed by previous students and continued in this thesis is the virtual sensor suite [71]. Despite its usefulness as a realistic abstraction of real sensors, the high number of individual sensors calls for a state-space reduction before feeding the data to the DRL algorithm. To do so, sensors are partitioned into sectors. How, then, should the distance information from a set of sensors be mapped to a single parameter for the sector? Two options are min and max pooling, which give overly pessimistic and optimistic estimates. Instead, the *feasibility pooling* procedure introduced in [71] is used. When applying this method, the maximum reachable distance within the sector is found, taking

the width of the vessel and the positions of any detected obstacles. The algorithm does this by iterating over the individual sensor readings belonging to the sector in question, from the shortest to longest detected distance, while determining whether or not the vessel can continue beyond this point without colliding. Two possible scenarios are shown in Fig. 2.4.1a.



(a) Full distance is reachable. **(b)** Less than half the distance is reachable.

Figure 2.4.1: Illustration of the feasibility algorithm for two different scenarios. The algorithm iterates over sensor readings in ascending order, and decides if the vessel can feasibly continue past this point. In the scenario displayed in the figure on the right, the opening is deemed too narrow for the full distance to be reachable [5].

The feasibility pooling algorithm can be found in Appendix A.

Chapter 3

Design and implementation

The overall goal of this thesis is to explore the incorporation of the COLREGs into an end-to-end and DRL-based path following and collision avoidance system. Since several other master students have contributed to the same line of research, a simulation environment was already in place. The elements of the environment essential to this work are described in Section 3.1, while details on the DRL algorithm itself, including the observation vector employed, are given in Section 3.3.

In line with the objectives presented in Section 1.3, a qualitative implementation of relevant COLREGs is explored in Section 3.4. This was done as it is the most commonly used method for reward shaping in RL. The work on the qualitative approach also served as a contribution to the collaborative paper Meyer et al. [5]. Next, a risk-based approach is taken in Section 3.5. Although quantification of collision risk is fuzzy, collision risk evaluation methods have historically been developed using statistical analysis and expert knowledge, as elaborated upon in Section 2.2.4. It is therefore a more quantitative approach, allowing for an interesting comparison with the former qualitative one.

3.1 Simulation environment structure

The simulation environment used in this work is based on [72], which in turn was developed using the OpenAI Gym [1] toolkit. OpenAI Gym is widely used for the development of RL algorithms. Moreover, the Python library Stable Baselines [2] provided the RL algorithm, PPO, itself. It is also worth mentioning that the

simulation of the rangefinder sensors is done with the help of the Shapely Python package [4], which allows for efficient calculation of intersection points once the objects in question (such as the OS and obstacles) are represented as geometric, planar shapes.

Furthermore, the simulation environment is modular, and consists of classes representing the main constituents of the system: environment, vessel, obstacles, path, and rewarder. The `Env` base class acts as a blank slate on which customisable elements such as vessels and obstacles can be added. Another way to see it is as an interface between the agent itself (the PPO algorithm) and the rest of the framework. For instance, the `Vessel` base class receives actions and sends observations through the `Env` class. In addition, instances of the `Obstacle` and `Path` base classes are implemented, which are also updated via `Env`. Finally, the `Rewarder` base class allows for straight-forward implementation of different reward schemes.

Specialised classes inheriting from the base classes can be implemented with ease. Two important classes, namely `MovingObstacles` and `RealWorld`, are examples of this. These environments inherit the `Env` properties and supply the simplified training scenario and AIS-based scenarios, respectively.

For a detailed account of the implementation and rendering details, please refer to [72], as only the overarching elements have been described here.

3.2 Vessel model

As outlined in Section 2.1, a 3-DOF model is employed, as it provides a minimal but sufficient model for the problem at hand. Specifically, the vessel dynamics of the miniature supply ship Cybership II depicted in Fig. 3.2 are used, with the parameters identified in [73]. This vessel is a 1:70 replica of length 1.225 m and mass 23.8 kg – a small and agile vessel. Further, it is equipped with rudders and propellers aft and one bow thruster fore, meaning it is a fully actuated vessel. For the sake of reducing the action space, however, the bow thruster is neglected. The resulting control vector is therefore $f_c = [f_u, f_r]^\top$, where f_u is the force input in surge, and f_r is the moment input in yaw.

Choosing a model of a smaller vessel with higher maneuverability allows the focus to remain on the DRL and COLREGs, as the training can be more challenging when using a more complex model that introduces destabilising characteristics such as inertial delay. In addition, using a smaller model means that testing on a real vessel would be relatively straight-forward later on.

Figure 3.2.1: A photo of the real-life CyberShip II vessel [74].

3.3 DRL algorithm details

The PPO algorithm described in Section 2.3.3.3 was chosen due to its ability to handle continuous action and state spaces and its convincing track records [75] [76]. After testing and comparing, it was found that a configuration of two hidden layers of 64 units each was sufficient for both the actor and the critic network. More complex or larger networks did not achieve better performance, but did generally require a longer time to converge, due to the higher number of parameters. Further, the hyperbolic tangent (tanh) activation function was used. The combination of small networks and the tanh activation function is default for PPO in Stable Baselines, and recommended in [75], supporting the choices. The hyperparameters used are summarised in Table 3.3.1.

Parameter	Interpretation	Value
γ	Discount factor	0.999
T	Timesteps per training iteration	1024
N_A	Number of parallel actors	8
K	Training epochs	10^6
η	Learning rate	0.0002
N_{MB}	Number of minibatches	32
λ	Bias vs. variance parameter	0.95
c_1	Value function coefficient	0.5
c_2	Entropy coefficient	0.01
ϵ	Clipping parameter	0.2

Table 3.3.1: Hyperparameters for PPO algorithm.

Furthermore, the vessel and sensor suite were configurated according to the values

in Table 3.3.2

Parameter	Interpretation	Value
U_{max}	Maximum vessel speed	10 m/s
N	Number of sensors	180
S_r	Sensor distance	1.5 km
d	Number of sensor sectors	9
Δ_{LA}	Look-ahead distance	3 km

Table 3.3.2: Vessel configuration

3.3.1 Observation vector

In order for the RL agent to learn a useful policy, it must be provided with sufficient information about the states of the vessel and the environment. An intuitive way to do so is by feeding it an observation vector s containing information relevant for path following and collision avoidance – which are the two main tasks. Considering *navigation* to be the identification of the vessel state relative to the desired path and *perception* the process of obtaining information through the sensor measurements, the problem can be broken down, such that $s = [s_n, s_p, \lambda]^T$, where $\lambda \in [0, 1]$ is a parameter aiding the weighting of the rewards given for the two tasks. In this section, the navigation-based and perception-based feature vectors s_n and s_p are described, while λ is described as a part of the reward function in Section 3.4.1.

3.3.1.1 Navigation

Whenever the vessel is not detecting obstacles closeby, it must efficiently follow the path. Therefore, it is necessary that the path navigation feature vector appropriately describes the vessel state in relation to the path. Using the parameterisation described in 2.2.1, the key features are those presented in Table 3.3.3. This way, the path-follwing feature vector becomes

$$s_n = \left[u, v, r, \epsilon, \tilde{\psi}, \tilde{\psi}_{LA}\right]^T \tag{3.3.1}$$

3.3.1.2 Perception

For the perception of its environment, the vessel is equipped with virtual rangefinder sensors. Since moving obstacles can approach on any side, the sensor suite is comprised of N distance sensors uniformly distributed around the vessel. These sensors have a maximum detection range of S_r.

Feature	Definition
Surge velocity	u
Sway velocity	v
Yaw rate	r
Cross-track error	ϵ
Heading error	$\tilde{\psi}$
Look-ahead heading error	$\tilde{\psi}_{LA}$

Table 3.3.3: Path following feature vector s_n.

Further, the individual sensors are assigned to D sectors according to a customisable distribution. Such partitioning was deemed necessary in [71], where it was noted that feeding raw sensor measurements to the agent leads to poor performance. A likely reason for this is a well-known challenge within AI research, namely the *curse of dimensionality*, which often arises when too many features are used and the agent is unable to generalise well [77].

One way to lower the dimensionality would be to simply decrease the number of sensors. However, this would also reduce the resolution – a highly undesirable situation. Instead, a logistic function is applied to the distribution of the sectors, reflecting the higher risk of collision associated with obstacles detected ahead. To achieve the logistic partitioning illustrated in Fig. 3.3.1, the sensors are mapped to sectors according to Eq. 3.3.2, where the index $i \in \{1, \ldots, N\}$ denotes the sensor index, and $\kappa \in \{1, \ldots, D\}$ denotes the sector index. Furthermore, σ denotes the logistic sigmoid function, while γ_C is a parameter allowing for adjustment of the sector distribution, ranging from uniform to denser in front of the vessel.

$$\kappa : i \mapsto \kappa(i) = \left\lfloor \underbrace{D\sigma\left(\frac{\gamma_C}{N} i - \frac{\gamma_C}{2}\right)}_{\text{Non-linear mapping}} - \underbrace{D\sigma\left(-\frac{\gamma_C}{2}\right)}_{\text{Constant offset}} \right\rfloor \tag{3.3.2}$$

Now that sensors have been assigned to sectors, the next step is mapping the sensor measurements to sector measurements. It is possible to simply return the minimum or maximum values, but these approaches are generally too conservative or liberal, either imposing unreasonable restrictions on the vessel's movements or allowing risky behaviour. In [71], a novel approach coined *feasibility pooling* was suggested, which takes the middle ground. This method is described in Section 2.4.1, and is applied here.

To ensure that all features are of the same magnitude, and to avoid discontinuities

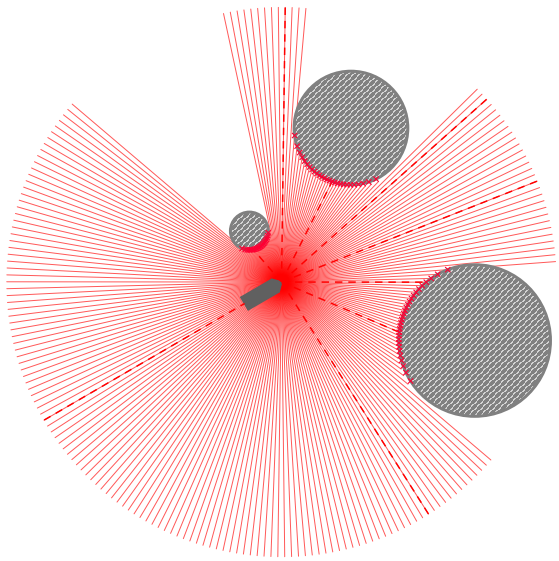


Figure 3.3.1: Rangefinder sensor suite partitioned into $D = 9$ sectors according to the the mapping function κ with the scale parameter $\gamma_C = 0.13$ [5].

in the input to the agent when an obstacle exits the sensor range, the distance d calculated by the feasibility pooling algorithm is converted to *closeness*. Defining this mapping as $c(d) : \mathbb{R} \mapsto [0, 1]$ such that 0 corresponds to no obstacle detection, and 1 to a collision, we have that

$$c(d) = \text{clip}\left(1 - \frac{d}{S_r}, 0, 1\right) \tag{3.3.3}$$

In addition to the rangefinder sensors, it is assumed that obstacle velocities are known. In real-life applications, this is usually obtained via AIS. To facilitate the agent's learning, the velocity vectors are decomposed relative to the center line of the corresponding sector. The y component is defined as parallel to the center line, and as positive in the direction towards the vessel. Conversely, the x component is defined normal to the center line, with positive values facing in the clockwise direction. This is illustrated in Fig. 3.3.2.

Finally, the velocity and position information obtained are concatenated, such that the perception-based feature vector becomes

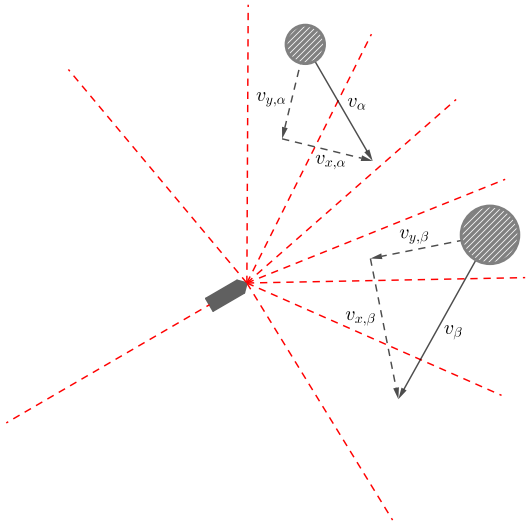


Figure 3.3.2: Velocity decomposition for two moving obstacles, α and β [5].

$$s_p = \left[\underbrace{c\left((d_1)\right), v_{x,1}, v_{y,1}}_{\text{First sector}}, \ldots, \right]^T \tag{3.3.4}$$

3.4 Qualitative implementation of COLREGs

To investigate the ability of DRL agents to learn COLREGs, it is interesting to first assess whether or not satisfactory behaviour can be learnt using a qualitative approach based on the already available rangefinder sensor and AIS information. In RL, a general goal is to keep the reward function as limited as possible, preventing over-engineering and allowing the agent to discover the optimal behaviour on its own. At the same time, such a bottom-up method is likely to introduce unexpected behaviours, since the RL agent may find local optimums when the reward function is too basic. Below, the rationale behind the qualitative reward function is presented, and the results are to be found in Section 3.4.

3.4.1 Reward function

Since the vessel's task is constituted by two well-defined subtasks, namely path following and collision avoidance, it makes sense to also define the reward function

in terms of these two subtasks, giving the independent terms r_{path} and r_{colav}. In essence, there is a trade-off between path following and collision avoidance, where the latter is considered most critical. At the same time, the vessel should focus on path following whenever the risk of collision is low. Therefore, a weighting coefficient $\lambda \in [0, 1]$ is used, as it was discovered that, depending on the tuning, the agent either too easily gave up on the path following task to avoid collisions, or disregarded the risk of collision when on the path. The weighting is done individually for each target vessel, which will be described in greater detail in Section 3.4.1.3.

Furthermore, to make sure that the vessel learns to move along the path rather than standing still, a living penalty $r_{exists} < 0$ is added to encourage completion of the path. Although the exact value of this penalty is unimportant, it must be sufficiently negative to prevent the agent from collecting positive reward while standing still. On the other hand, it cannot be excessively negative to the point where it disrupts learning of path following and collision avoidance. Setting $r_{exists} = -1$ yielded the desired balance.

In addition, it is crucial that the agent receives a sufficiently negative reward $r_{collision}$ whenever a collision is detected. Combining the elements presented so far, the tentative and simplified reward becomes

$$r = \begin{cases} r_{collision}, & \text{if collision} \\ \lambda r_{path} + (1 - \lambda)\, r_{colav} + r_{exists}, & \text{otherwise} \end{cases} \tag{3.4.1}$$

Since we are dealing with both static and dynamic obstacles, which pose inherently different risks, the reward r_{colav} is further divided into two parts, $r_{colav,stat}$ and $r_{colav,dyn}$. This way, it is possible to regard the COLREGs only when dealing with dynamic obstacles (other vessels), and to adjust the penalties given when detecting static and dynamic obstacles such that the respective risk levels can be accounted for. This gives

$$r_{colav} = \underbrace{r_{colav,stat}}_{\text{Static component}} + \underbrace{r_{colav,dyn}}_{\text{Dynamic component}} \tag{3.4.2}$$

In the following subsections, the path following reward and the static and dynamic collision avoidance rewards are described in detail. Instead of dividing the configuration of the parameters into individual tables for each section, the complete configuration is provided here, in Table 3.4.1, for reference.

Parameter	Interpretation	Value
γ_e	Cross-track error scaling	0.5
α_x	Raw COLAV penalty scaling	75
$\gamma_{\theta,stat}$	Sensor angle scaling for static obst.	10
$\gamma_{\theta,dyn}$	Sensor angle scaling for dyn. obst.	1
γ_x	Static obstacle distance scaling	0.01
$\gamma^+_{v,st.b.}$	Scaling of $v_y \geq 0$, s.b. side	0.004
$\gamma^-_{v,st.b.}$	Scaling of $v_y < 0$, s.b. side	0.05
$\gamma^+_{v,port}$	Scaling of $v_y \geq 0$, port side	0.007
$\gamma^-_{v,port}$	Scaling of $v_y < 0$, port side	0.005
$\gamma^+_{v,stern}$	Scaling of $v_y \geq 0$, astern	0.007
$\gamma^-_{v,stern}$	Scaling of $v_y < 0$, astern	0.005
$\gamma_{x,st.b.}$	Dyn. obst. distance scaling, st.b.	0.007
$\gamma_{x,port}$	Dyn. obst. distance scaling, port	0.009
$\gamma_{x,stern}$	Dyn. obst. distance scaling, stern	0.01
α^+_λ	Translation of λ, $v_y \geq 0$	4
α^-_λ	Translation of λ, $v_y < 0$	2
γ^+_λ	Distance scaling of λ, $v_y \geq 0$	0.003
γ^-_λ	Distance scaling of λ, $v_y < 0$	0.005
r_{coll}	Collision reward	-10000
r_{exists}	Living penalty	-1

Table 3.4.1: Reward configuration for the qualitative approach

3.4.1.1 Path following performance

The overall behaviour classified as path following can be broken down to two distinct behaviours: 1) reduction of distance to the path, and 2) movement along the path. Doing this allows for efficient incentivisation based on the navigation features provided as input to the agent.

One way to formalise the reward given when the agent reduces the distance to the path is through the cross-track error (CTE), as the absolute CTE $\left|\epsilon^{(t)}\right|$ is equivalent to the distance to the path. The next question is how to design the reward function such that it gives rise to the desired behaviours in the agent. As discussed in Section 2.3.2.3, there problem of vanishing gradient is likely to occur if the reward is too sparse. At the same time, it is necessary to accentuate the reward given when the agent is behaving correctly. The exponential function of form $\exp\left(-\gamma_\epsilon \left|\epsilon^{(t)}\right|\right)$ with $\gamma_\epsilon > 0$ combines a sharp peak for small $\left|\epsilon^{(t)}\right|$ and flat tails for more efficient learning, and is therefore chosen here. It should be mentioned that a regular Gaussian is generally considered the go-to function when penalising cross-track error in simple path following [78] [37]. Path following combined with

collision avoidance requires a slightly different function profile, however, as the agent must retain its overall path following goal while straying from it to avoid collisions. As shown in Fig. 3.4.1a, to achieve flat tails similar to those present in the absolute exponential, the Gaussian requires a large and relatively flat area for small cross-track errors, leading to imprecise rewards in this region.

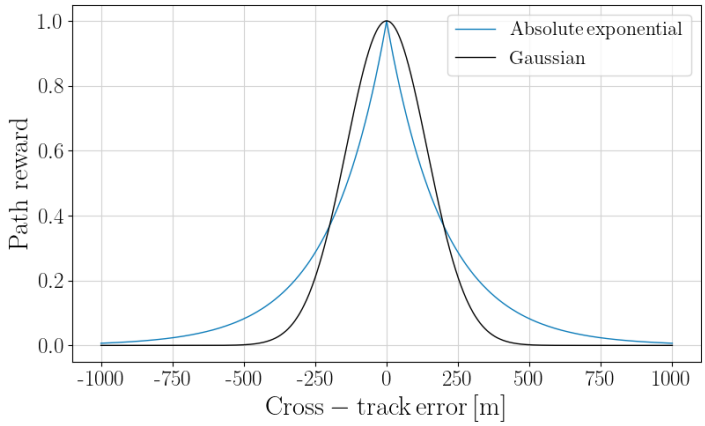
For the second desired behaviour, progression along the path, the heading error $\tilde{\psi}$ provides a useful basis. Since we would like to reward the agent such that the maximum reward is given when the heading error is zero and the speed is equal to the maximum vessel speed U_{max}, the term $\frac{u^{(t)}}{U_{max}} \cos \tilde{\psi}^{(t)}$ was chosen. This also means that a negative reward is given if the OS is travelling in the opposite direction to the path. In order to combine the two terms derived for CTE reduction and path tracking, they are simply multiplied. Since the CTE-based term is always positive and the velocity-based term takes values in the range $[-1, 1]$, the properties described are preserved, and the preliminary path following reward becomes

$$r_{path}^{(t)} = \underbrace{\frac{u^{(t)}}{U_{max}} \cos \tilde{\psi}^{(t)}}_{\text{Velocity-based reward}} \underbrace{\exp\left(-\gamma_\epsilon |\epsilon^{(t)}|\right)}_{\text{CTE-based reward}}$$

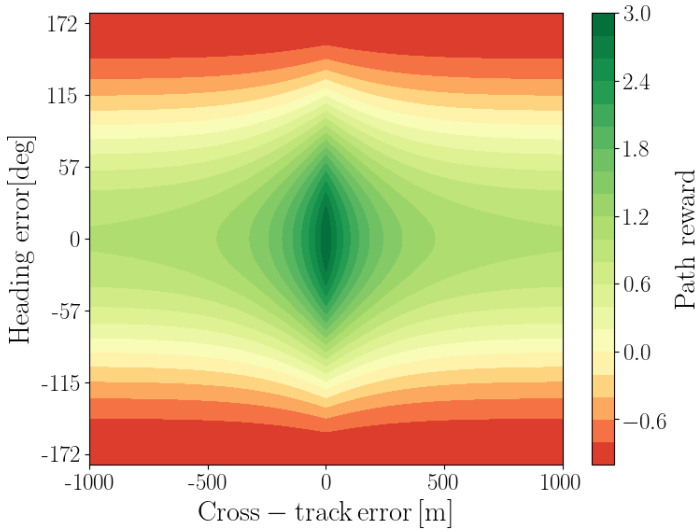
A problem with the reward function above is the presence of corner cases where the reward is zero. This happens whenever the vessel is immobile and $u^{(t)} = 0$, when moving at an angle perpendicular to the path such that $\tilde{\psi}^{(t)} = \pm\frac{\pi}{2}$, and when the absolute CTE becomes large such that $\exp\left(-\gamma_\epsilon \left|\epsilon^{(t)}\right|\right) \to 0$. To resolve the two former situations, constant multiplier terms γ_r are added to each term to ensure nonzero rewards in the aforementioned situations. Doing so results in a constant reward bias, which is then removed by subtracting γ_r^2. This gives the final reward function for path following

$$r_{path}^{(t)} = \underbrace{\left(\frac{u^{(t)}}{U_{max}} \cos \tilde{\psi}^{(t)} + \gamma_r\right)}_{\text{Velocity-based reward}} \underbrace{\left(\exp\left(-\gamma_\epsilon |\epsilon^{(t)}|\right) + \gamma_r\right)}_{\text{CTE-based reward}} - \gamma_r^2 \tag{3.4.3}$$

Since the multiplication of the two terms leads to interdependence and a more complex reward function, it could be argued that it would be better to simply add the terms together instead. With the proposed solution, however, the two terms are linked in such a way that there is a significantly higher reward when both aspects of the path following (CTE and velocity) are fulfilled than when only one is. This accentuated difference is visible in Fig. 3.4.1b.



(a) Cross-section of the path following reward landscape assuming path-tangential full-speed motion visualized for both Gaussian and absolute exponential kernels for cross-track error rewarding.



(b) Path following reward function assuming full speed.

Figure 3.4.1: Cross-section and level curves for the path following reward function with $\gamma_\epsilon = 0.05$ [5].

3.4.1.2 Static obstacle avoidance performance

Moving onto reward shaping for static obstacle avoidance, the requirements are largely opposite; instead of being awarded with a positive reward when moving closer, the reward should be increasingly negative. To penalise a decreasing distance, or increasing closeness, a raw distance penalty is defined based on the distance x measured by the sensor in question. This way, the sensor measurements are used directly in the calculation of the reward.

Since the rate of change of the collision risk is intuitively much higher as the OS is getting closer to a TS, it is natural to employ a nonlinear function for the raw distance penalty. For instance, an exponential function accurately reflects how a decrease in distance from the TS of a few metres is significantly worse when the TS is already close. A good function candidate is therefore $\alpha_x \exp(-\gamma_x x)$, where α_x and γ_x are chosen such that sufficiently high negative rewards are given as objects get closer to the OS, but no negative reward is given when the shore or object is considered to be at a safe distance. The tuning of these parameters is highly dependent on the type of situation at hand, as the accepted shortest distance from the OS to an obstacle varies greatly according to the terrain. Here, values were chosen such that negative rewards are given when the distance measured is less than 500 m, and falls sharply when entering a radius of 250 m – as shown in Fig. 3.4.2.

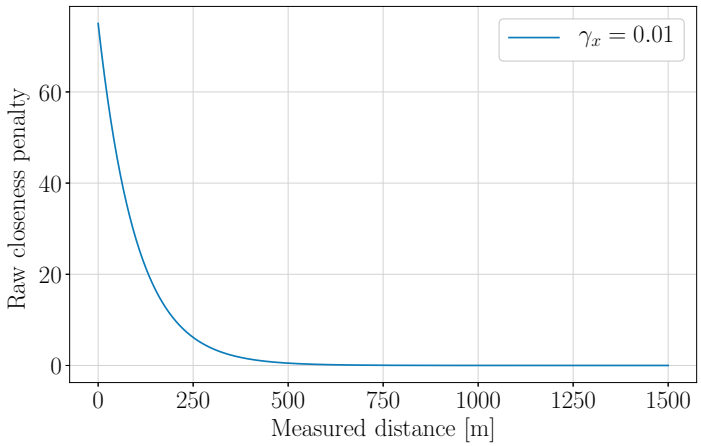


Figure 3.4.2: Plot showing the raw distance penalty for static obstacles with $\alpha_x = 75$.

Moreover, it is useful to consider the immobility of static obstacles to limit the reactivity of the agent. Provided that no external forces such as wind or current are acting on the OS, and that the vessel is unable to move backwards, a static obstacle only poses a threat when detected ahead of the OS. By adding a weighting term, this can be integrated into the path following reward function. A function that highly prioritises the measurements taken by sensors associated with small angles relative to the centerline was chosen, and is shown in blue in Fig. 3.4.3. The weighting term for dynamic obstacles is also shown, and will be dicussed in the next section.

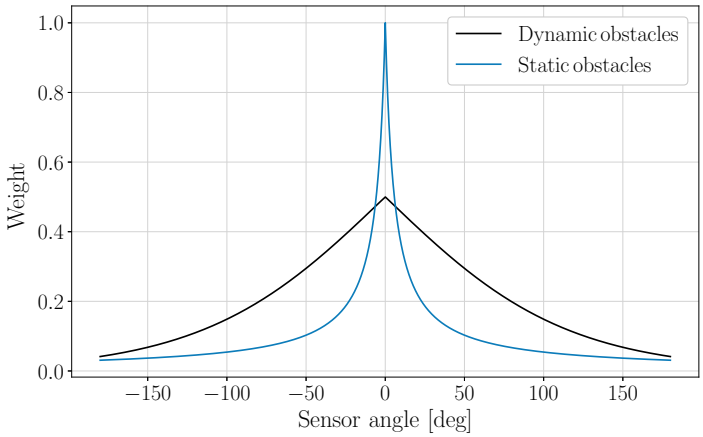


Figure 3.4.3: Plot showing sensor angle weighting for static and dynamic obstacles

The resulting reward given when a static obstacle is detected at a distance x, and at a sensor angle θ with respect to the centerline of the vessel is therefore given by

$$r_{obst,stat}^{(t)} = -\underbrace{\frac{1}{1+\gamma_{\theta,stat}|\theta|}}_{\text{Weighting term}} \underbrace{\alpha_x \exp\left(-\gamma_x x\right)}_{\text{Raw distance penalty}} \tag{3.4.4}$$

For the entire sensor suite, the overall collision avoidance reward for static obstacles, $r_{colav,stat}$, is then given as the weighted average of the individual rewards. Denoting the i^{th} distance sensor measurement as x_i and the vessel-relative angle of the corresponding sensor ray as θ_i, the overall reward becomes as in Eq. 3.4.5. A weighted average is used to prevent the weighting term itself from introducing

effects that are unaccounted for, as the only objective is to place assign relative priorities. For instance, the weight distributions used for static and dynamic obstacles can now be compared without considering the exact values of the weights.

$$r_{colav,stat}^{(t)} = -\frac{\sum_{i=1}^{N} \frac{\alpha_x}{1+\gamma_{\theta,stat}|\theta_i|} \exp\left(-\gamma_x x_i\right)}{\sum_{i=1}^{N} \frac{1}{1+\gamma_{\theta,stat}|\theta_i|}} \tag{3.4.5}$$

In Fig. 3.4.4, the static collision avoidance penalty is illustrated as a function of obstacle distance and angle. As expected, the highest penalty (most negative reward) is assigned at small distances and angles.

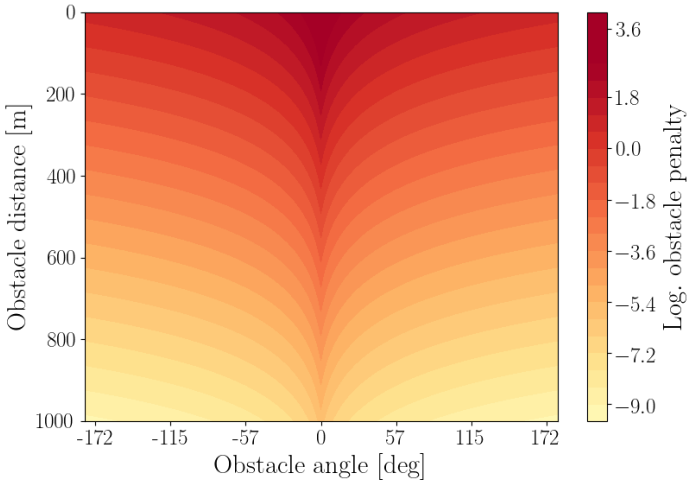


Figure 3.4.4: Static obstacle closeness penalty landscape as a function of obstacle distance and angle relative to the vessel with the scale parameters $\gamma_\theta = 10$, $\gamma_x = 0.01$. The maximum penalty is imposed for obstacles located right in front of the vessel [5].

3.4.1.3 Dynamic obstacle avoidance performance

For dynamic obstacle avoidance, we expand on the framework developed for static obstacles. Firstly, the penalty should reflect the relevant COLREGs such that the agent is incentivised to learn COLREG-compliant behaviour. Since the COLREGs

are defined according to the bearing of a TS relative to the OS, an intuitive way to guide the RL agent towards COLREGs compliance is to adjust the static obstacle penalty (Eq. 3.4.4) according to the relative bearing of the dynamic obstacle. The area around a vessel is normally split into three sectors: port, starboard, and stern, as illustrated in Fig. 3.4.5. Therefore, a tunable parameter ζ_x was added to allow for differentiated weighting of these sectors.

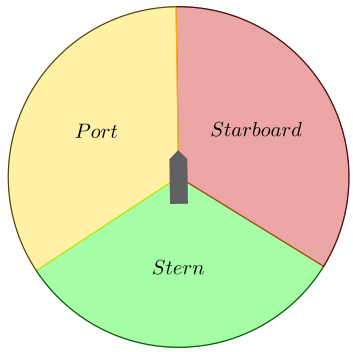


Figure 3.4.5: Illustration of sectors around the OS [5].

According to the COLREGs, it is desirable that crossings take place on the port side, meaning that the weighting of sensor readings on the starboard side should be higher. However, since it is assumed at this stage of the work that the target vessels have restricted maneuverability, sensor readings on the port side and astern must also be sufficiently penalized, such that the OS acts as the give-way ship in all situations. Since the reward function is such that a lower value of γ_x gives a higher penalty (more negative reward), denoting starboard as "st.b.", we thus have that $\gamma_{x,st.b.} < \gamma_{x,port} \leq \gamma_{x,stern}$.

$$\zeta_x(\theta) = \begin{cases} \gamma_{x,st.b.}, & \text{if } \theta \geq 0° \text{ and } \theta < 112.5° \\ \gamma_{x,port}, & \text{if } \theta \geq -112.5° \text{ and } \theta < 0° \\ \gamma_{x,stern}, & \text{if } \theta \geq 112.5° \text{ or } \theta < -112.5° \end{cases} \tag{3.4.6}$$

Furthermore, the reward must reflect the variable risk associated with the direction of a TS's velocity; an approaching TS gives rise to a much higher risk than a receding one. In addition, the relatively steep function used as weighting term in Eq. 3.4.4 was exchanged for a flatter function of the form $1/(1 + \exp(x))$, so as to give dynamic obstacles detected around the OS sufficient priority. This is showcased in Fig. 3.4.3. It should be re-iterated that it is not the numerical value

of the weights that matters, but rather the relative difference between them. Since the static and dynamic obstacles are treated separately, the weights distributed among static obstacles do not affect the dynamic obstacles, and vice versa. Making adjustments to the static obstacle penalty to adhere to the requirements discussed, the penalty for a single dynamic obstacle was chosen as

$$r_{obst,dyn} = -\underbrace{\frac{1}{1+\exp(\gamma_{\theta,dyn}|\theta|)}}_{\text{Weighting term}} \underbrace{\alpha_x \exp\left((\zeta_v v_y - \zeta_x)x\right)}_{\text{Raw penalty}} \tag{3.4.7}$$

where x is the distance to the obstacle, θ is the vessel-relative angle (bearing angle), and v_y is the velocity component in the direction towards the vessel. The scaling factor for velocity, ζ_v, is given as a function of the angle θ and the velocity v_y, whilst the scaling factor for distance ζ_x is a function of the angle θ only. Such a shaping of the reward function parameters acts as in indirect classification of encounter situations and guides the agent towards COLREG-compliant behavior. As was mentioned, it is often argued that reward functions in RL should not be over-engineered, which might seem to be the case here. However, it was found that an algorithm with less adaptation of the reward function according to the situation, and therefore fewer parameters, was in fact harder to tune due to the subsequent high level of dependency between different encounter situations. Due to these difficulties, it was deemed impossible to achieve COLREGs-compliance with fewer adjustable parameters in this setup. For instance, since the starboard side is already heavily penalised, lighter weighting of velocity was needed to prevent the agent from reacting too strongly when detecting a TS on the starboard side. The sign of the velocity component of the TS towards the OS is therefore used to determine whether the TS is moving towards the OS or moving away, which together with the sensor angle θ provides a good basis for determining a reasonable scaling factor for v_y. This scaling factor, ζ_v, is therefore given as

$$\zeta_v(\theta, v_y) = \begin{cases} \begin{cases} \gamma^+_{v,st.b.} & \text{if } v_y \geq 0 \\ \gamma^-_{v,st.b.} & \text{if } v_y < 0 \end{cases} & \text{if } \theta > 0° \text{ and } \theta < 112.5° \\ \begin{cases} \gamma^+_{v,port} & \text{if } v_y \geq 0 \\ \gamma^-_{v,port} & \text{if } v_y < 0 \end{cases} & \text{if } \theta > -112.5° \text{ and } \theta < 0° \\ \begin{cases} \gamma^+_{v,stern} & \text{if } v_y \geq 0 \\ \gamma^-_{v,stern} & \text{if } v_y < 0 \end{cases} & \text{otherwise} \end{cases} \tag{3.4.8}$$

An illustration of the effects of ζ_x and ζ_v on the overall reward for each sensor is presented in Fig. 3.4.6, where the black and blue graphs represent the baseline case where a dynamic obstacle is detected with zero speed and the grey and red graphs represent a detected obstacle speed of $v_y = 5$ ms^{-1} – a speed close to the average of the target vessels included from the AIS-based scenarios. A few different configurations were tested for the reward term $r_{obst,dyn}$, with extensive tuning each time. Initially, for instance, the velocity term was added such that the raw distance penalty was given as $\alpha_x \exp(\zeta_v v_y - \zeta_x x)$, which appeared more natural. However, this function decreases to zero more or less at the same point regardless of the detected speed v_y. The final form was chosen due to the way a positive detected speed "lifts" the function upwards. This ensures that the penalty is negative for a larger radius around the OS, without significantly altering the characteristics of the reward function.

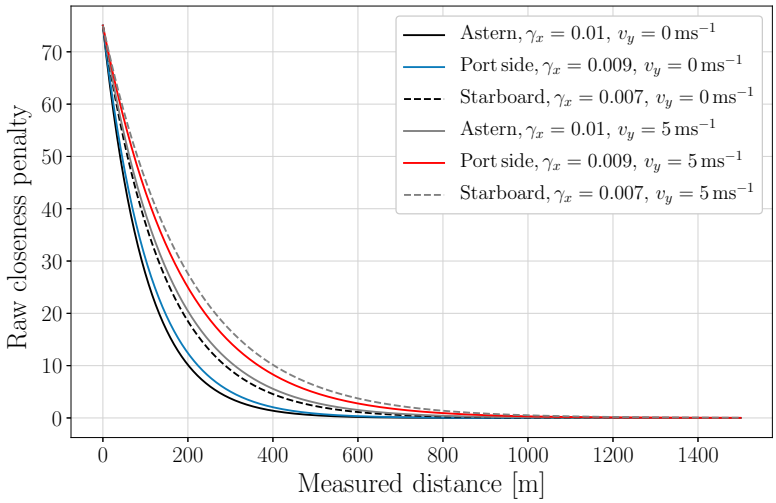


Figure 3.4.6: Plot showing the raw distance penalty for dynamic obstacles with $\alpha_x = 75$.

Finally, as was done for static obstacles, we then compute the complete dynamic

obstacle avoidance reward according to the weighted average

$$r_{colav,dyn}^{(t)} = -\frac{\sum_{i=1}^{N} \frac{\alpha_x(1-\lambda_i)}{1+\exp(\gamma_{\theta,dyn}|\theta_i|)} \exp\left((\zeta_v v_y^i - \zeta_x)x_i\right)}{\sum_{i=1}^{N} \frac{1}{1+\exp(\gamma_{\theta,dyn}|\theta_i|)}} \tag{3.4.9}$$

where λ_i is a parameter referred to as a *prioritisation factor*. The prioritisation factor regulates the relative importance of path following and collision avoidance in an encounter situation. A higher value indicates a greater focus on path following, while a low value ignores the path following in favor of collision avoidance. This parameter function depends on the velocity v_y^i detected, and takes the distance x_i measured by the sensor as input, according to the logistic function

$$\lambda_i^{(t)} = \frac{1}{1+\exp\left(-\gamma_\lambda(v_y^i)x_i^{(t)} + \alpha_\lambda(v_y^i)\right)} \tag{3.4.10}$$

Here, $\alpha_\lambda(v_y)$ and $\gamma_\lambda(v_y)$ are tunable parameters. Two sets of constant values were chosen such that the overall function for λ_i would depend solely on the sign of the speed v_y of the TS towards the OS, giving higher values when $v_y < 0$. In other words, λ_i incorporates the difference in risk between crossing ahead and astern of a TS, allowing the agent to return to path following quicker in a situation where the TS facing away from the OS. Formally, we have

$$\alpha_\lambda(v_y) = \begin{cases} \alpha_\lambda^+, & \text{if } v_y \geq 0 \\ \alpha_\lambda^-, & \text{if } v_y < 0 \end{cases} \tag{3.4.11}$$

and

$$\gamma_\lambda(v_y) = \begin{cases} \gamma_\lambda^+, & \text{if } v_y \geq 0 \\ \gamma_\lambda^-, & \text{if } v_y < 0 \end{cases} \tag{3.4.12}$$

The difference in the prioritisation factor in the respective scenarios ($v_y \geq 0$ and $v_y < 0$) is illustrated in Fig. 3.4.7. In the case of a receding target vessel, path following is prioritised higher than collision avoidance whenever the distance exceeds approximately 400 m. Conversely, this only happens at approximately 1300 m in the case of an approaching target vessel.

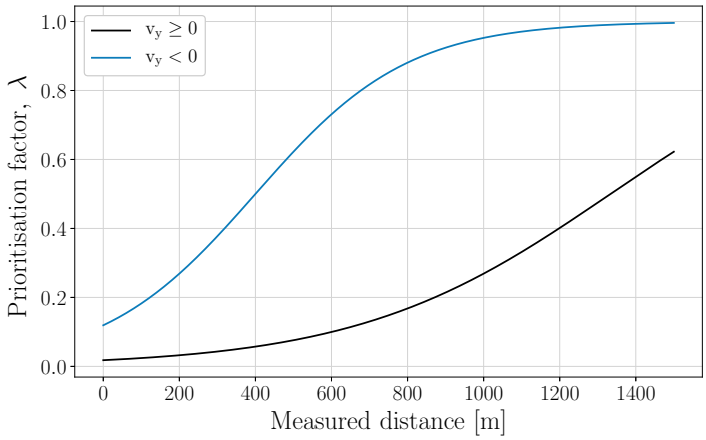


Figure 3.4.7: Plot showing the prioritisation factor λ as a function of the measured distance x, for $v_y \geq 0$ and $v_y < 0$.

3.5 Risk-based implementation of COLREGs

A risk-based approach offers an interesting contrast to the qualitative implementation elaborated on in Section 3.4, and could provide a useful leverage point in future research. Building on the theory presented in Section 2.2.4, a collision risk index (CRI) is calculated using fuzzy evaluation. Here, this translates to a weighted sum of evaluated risk factors, a method which is described in detail in Section 3.5.1.1. Doing so elegantly encapsulates the continuous and fuzzy nature of collision risk, making it a convincing choice for translating the COLREGs into a DRL-based framework.

3.5.1 Reward function

Collision risk is normally only applied to encounter situations between two dynamic objects, and the collision risk index to be presented here is no exception. This means that the components for path following, static obstacle avoidance, collision penalty and living penalty must be defined separately. Reusing the corresponding parts from the previous approach makes sense since the DRL setup for path following and static obstacles produced excellent results in the earlier stages of the work. The rewards for path following and static obstacle avoidance are given in Eq. 3.4.3 and Eq. 3.4.5 respectively, while the collision and living penalties are negative integers.

As a result, the total reward function has the same structure as before, reiterated below, only with a risk-based penalty for dynamic obstacles ($r_{colav,dyn}$).

$$r = \begin{cases} r_{collision}, & \text{if collision} \\ \lambda r_{path} + (1-\lambda)\, r_{colav} + r_{exists}, & \text{otherwise} \end{cases} \tag{3.5.1}$$

The penalty for dynamic obstacles makes part of the overall penalty for collision avoidance, denoted r_{colav} and given by

$$r_{colav} = r_{colav,dyn} + r_{colav,stat} \tag{3.5.2}$$

For every TS detected by the OS, a collision risk index (CRI) $\in [0, 1]$ is calculated (see Section 3.5.1.1). Due to the index's property of increasing with increasing collision risk, it can be used semi-directly in the reward function. By multiplying the CRI_i of each target vessel i with a scaling factor $\beta_{CRI} > 0$, the penalty level can be adjusted to the right magnitude relative to the rest of the reward function:

$$r_{colav,dyn} = -\sum \beta_{CRI} \cdot CRI_i \tag{3.5.3}$$

3.5.1.1 Calculation of collision risk index

In order to determine the collision risk in an encounter situation, one must first define what constitutes a collision risk, and how much each risk factor contributes to the overall risk. The state-of-the-art methods of computing CRI generally make use of fuzzy evaluation to do so [14], which is therefore a natural choice here. In short, three steps should be followed:

1. Define individual risk factors
2. Define membership functions
3. Design overall CRI as a function of membership functions

In the following, the chosen risk factors and their membership functions are elaborated on, before designing the CRI function based on them.

A common starting point for defining risk is looking at the distance and time to the point of closest approach, denoted DCPA and TCPA. As the explanatory name reveals, the closest point of approach (CPA) is the closest point relative to the OS that the TS in question will come, given that the relative course and relative velocity between the two ships stay the same. The DCPA, then, is the distance to the CPA, whilst the TCPA is the time until the TS arrives at the CPA. Put differently, the DCPA quantifies the severity of a potential collision situation, while the TCPA quantifies the urgency of it. It is customary to employ upper and lower bounds for these quantities when determining the risk level associated with them, denoted d_L and d_U for DCPA, and t_L and t_U for TCPA. Doing so, the membership functions u_{DCPA} and u_{TCPA} output 1 (highest risk level) whenever |DCPA| $\leq d_L$ and |TCPA| $\leq t_L$, respectively. Conversely, the outputs are 0 when |DCPA| $\geq d_U$ and |TCPA| $\geq t_U$. As was done in Gang et al. [79], a second-order function is used between the two extremities. In other works, such as Chen et al. [80], a sinusoidal function is used instead. Although the latter has the virtue of being smooth, it was deemed to be inexpedient due to the large outputs for a wide interval of values, overshadowing other elements of the CRI. Since the sensor range used in this work is relatively short (1500 m), the steeper second-order function was found to improve learning. It is worth noting that the sinusoidal function may be more suited in a setup with fewer obstacles and vessels where AIS data from a larger region is used.

The values for the lower and upper bounds depend largely on the application. In general, d_L defines the minimal safe encounter distance and d_U is the absolute safe encounter distance [79]. For DCPA, we thus have that

$$u_{DCPA} = \begin{cases} 1 & \text{if } |DCPA| \leq d_L \\ 0 & \text{if } |DCPA| \geq d_U \\ \left(\frac{d_U - |DCPA|}{d_U - d_L}\right)^2 & \text{else} \end{cases} \tag{3.5.4}$$

with d_L and d_U as positive integers. The DCPA membership function is presented graphically in Fig. 3.5.1.

For the bounds on TCPA, the method used in [79] and presented in Eq. 3.5.5 is employed. Doing so adjusts the output of u_{TCPA} according to the distance between the OS and TS, accurately presenting the high risk when the distance is below or close to the lower bound d_L and low risk when it is closer to the upper bound d_L. It should be noted that it is assumed that DCPA never exceeds d_U, meaning that d_U is set to the maximum detectable DCPA.

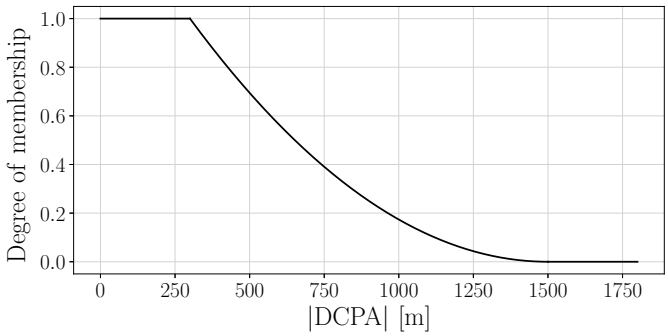


Figure 3.5.1: Membership function for DCPA with $d_L = 320$ m and $d_U = 1500$ m.

$$t_L = \begin{cases} \frac{\sqrt{d_L^2 - DCPA^2}}{v_R} & \text{if } |DCPA| \leq d_L \\ \frac{d_L - DCPA}{v_R} & \text{if } |DCPA| > d_L \end{cases} \tag{3.5.5a}$$

$$t_U = \frac{\sqrt{d_U^2 - DCPA^2}}{v_R} \tag{3.5.5b}$$

In [79], equal importance has been given to positive and negative values of TCPA, through the membership function for TCPA below:

$$u_{TCPA} = \begin{cases} 1 & \text{if } |TCPA| \leq t_L \\ 0 & \text{if } |TCPA| \geq t_U \\ \left(\frac{t_U - |TCPA|}{t_U - t_L}\right)^2 & \text{else} \end{cases} \tag{3.5.6}$$

However, noting that negative values of TCPA indicate that the OS and TS have passed each other, it makes sense to pay attention to the sign of TCPA. This is supported by [81], where a fuzzy case-based reasoning system for collision avoidance is proposed. In their work, the TCPA membership function in Fig. 3.5.2 is employed, indicating the significantly higher risk associated with positive values of TCPA.

Following this line of reasoning, a distinction between positive and negative values of TCPA is made according to Eq. 3.5.7. The cut-off value for negative values (negative limit) was chosen as $t_{NL} = \frac{d_L}{v_R}$, such that the degree of membership is

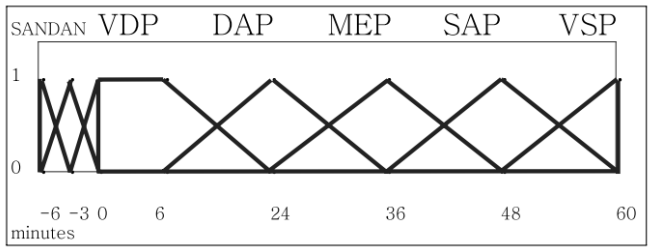


Figure 3.5.2: Membership function for TCPA employed in [81]. SAN = Safe Negative, DAN = Dangerous Negative, VDP = Very Dangerous Positive, DAP = Dangerous Positive, MEP = Medium Positive, SAP = Safe Positive, VSP = Very Safe Positive.

larger than zero whenever the OS is less than d_L time steps away from the TS.

$$u_{TCPA} = \begin{cases} \begin{cases} 1 & \text{if } TCPA \leq t_L \\ 0 & \text{if } TCPA \geq t_U \\ \left(\frac{t_U - TCPA}{t_U - t_L}\right)^2 & \text{else} \end{cases} & \text{if } TCPA \geq 0 \\ \begin{cases} 0 & \text{if } TCPA \leq t_L \\ \left(\frac{t_{NL} - |TCPA|}{t_{NL}}\right)^2 & \text{else} \end{cases} & \text{if } TCPA < 0 \end{cases} \tag{3.5.7}$$

The membership function for TCPA is plotted in Fig. 3.5.3.

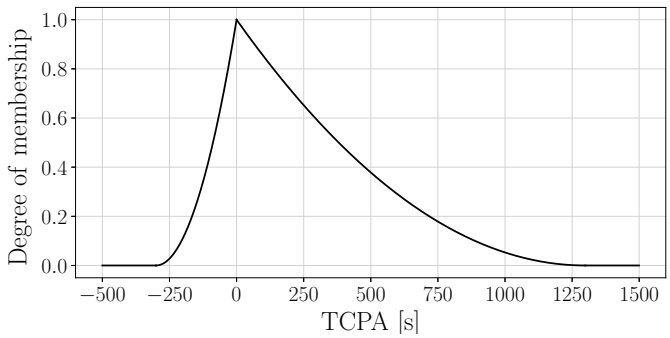


Figure 3.5.3: Membership function for TCPA with $d_L = 320$ m, $d_U = 1500$ m and $v_R = 1$ m/s.

Further, the collision risk depends on the position of the TS relative to the OS, which can be expressed through the absolute distance R between them and the

bearing angle of the TS, θ_T. Since the risk is higher on the starboard side of the OS, as expressed in Rule 14 (head-on situation) of the COLREGs, the membership functions should be designed with a bias on that side. Inspired by Davis et al. [58], it is customary to introduce a bias of 19° starboard. Davis developed the concept of ship arena, briefly described in Section 2.2.4, and designed a scaling of the upper bound as

$$R_D = 1.7 \cos\left(\theta_T \frac{\pi}{180} - 19°\right) + \sqrt{\left(4.4 + 2.89 \cos^2\left(\theta_T \frac{\pi}{180} - 19°\right)\right)} \tag{3.5.8}$$

while the lower bound is usually taken as 12 times the OS length L_{pp} [79], but set to $8L_{pp}$ here due to smaller scale. Initially, the upper bound given by R_D was implemented, but it quickly became clear that adjustments had to be made to make sure the agent received sufficiently negative reward when approaching TSs, regardless of their bearing angle. The difference in scaling of 4.4 times for ships detected at 19° and 161 ° (180°−19°) was too large considering the relatively densely populated training and testing scenarios and a restricted sensor range of 1500 m. Through testing, it was observed that the distance membership function could be made uniform while still preserving the correct behaviour in head-on situations as long as the membership function for the bearing angle θ_T was given enough weight. As a result, the lower and upper bounds for the absolute distance R were chosen as

$$R_L = \beta_{RL} L_{pp} \tag{3.5.9a}$$

$$R_U = \beta_{RU} L_{pp} \tag{3.5.9b}$$

with β_{RL} and β_{RU} chosen as appropriate scaling constants.

Following the logic applied to the membership functions for TCPA and DCPA, we arrive at the following membership function for the absolute distance between the OS and TS:

$$u_R = \begin{cases} 1 & \text{if } R \leq R_L \\ 0 & \text{if } R \geq R_U \\ \left(\frac{R_U - R}{R_U - R_L}\right)^2 & \text{else} \end{cases} \tag{3.5.10}$$

To ensure the appropriate behaviour in head-on situations, the function for the

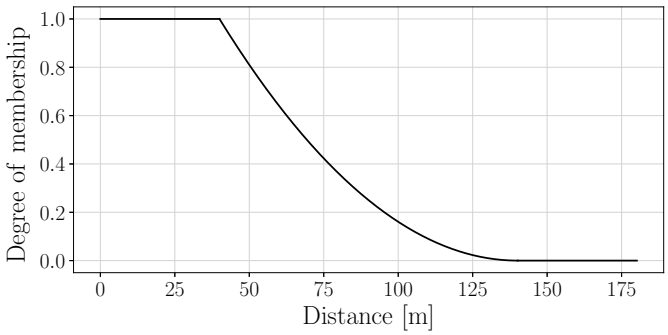


Figure 3.5.4: Membership function for distance to the target ship, with $\theta_T = 0°$.

bearing angle of the TS relative to the OS should be largest on the starboard side. Defining θ_{PU}, θ_{PU}, θ_{PU}, and θ_{PU} as the positive upper, positive lower, negative upper, and negative lower bounds on θ_T, the membership function for the bearing angle can be defined as below and illustrated in Fig. 3.5.5.

$$u_{\theta_T} = \begin{cases} \text{clip}\left(\left(\frac{\theta_{PU}-\theta_T}{\theta_{PU}-\theta_{PL}}\right)^2, 0, 1\right) & \text{if } \theta_T \geq 0 \\ \text{clip}\left(\left(\frac{\theta_{NU}-|\theta_T|}{\theta_{NU}-\theta_{NL}}\right)^2, 0, 1\right) & \text{if } \theta_T < 0 \end{cases} \tag{3.5.11}$$

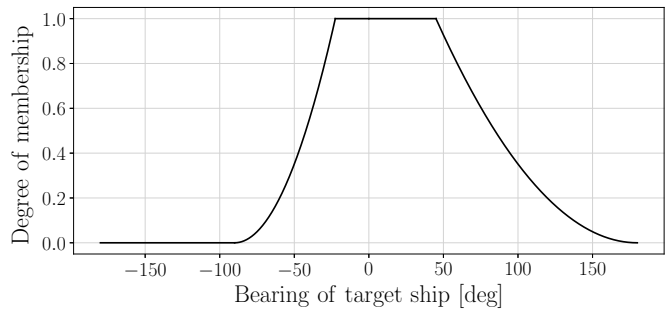


Figure 3.5.5: Membership function for the bearing angle of the target ship, with $\theta_{PU} = 180°$, $\theta_{PL} = 45°$, $\theta_{NU} = 90°$, and $\theta_{NL} = 22.5°$.

After implementing a CRI containing the four membership functions introduced so far, it became clear that it was necessary to add an element to the CRI to deter the OS from crossing ahead of a TS. Since the speed of the TS towards the OS can

be used to quantify whether or not the OS is ahead of the TS, and the speed of the TS towards and perpendicular to the OS (v_y and v_x) are readily available in the observation vector, an additional membership function was designed based on these speeds. Hence, we define $u_V(\cdot)$ as the ratio of the TS's speed towards the OS to its absolute speed, as in Eq. 3.5.12. The choice of such a ratio was done to avoid issues with differences in speed among the TSs, which easily could have arisen if the numerical value of v_y had been used instead. On the other hand, it might be desirable to distinguish between crossing ahead ships travelling at different speeds, as faster ships naturally pose a higher risk. However, since the speeds of the TSs in the training and testing scenarios have been sampled from a narrow range of 6 ± 1 m/s, it is considered outside of the scope of this work. It is worth noting that $u_V(\cdot)$ is negative when v_y is negative, emphasising the advantage of astern crossings.

$$u_V = \frac{v_y}{\sqrt{v_x^2 + v_y^2}} \tag{3.5.12}$$

Integrating the introduced membership functions into a collision risk index, we have that

$$CRI = \max\left(0, \alpha_{CPA}\sqrt{u_{DCPA} \cdot u_{TCPA}} + \alpha_{\theta_T} u_{\theta_T} + \alpha_R u_R + \alpha_V u_V\right) \tag{3.5.13}$$

where the CPA composite term was designed in such a way that a combination of low values for both DCPA and TCPA gives rise to a high CRI. It also accurately expresses how a low value of either DCPA or TCPA significantly reduces the overall risk. The `max`-function is applied to ensure that the CRI is always larger or equal to zero.

Finally, values are assigned to the weights such that the sum is equal to unity, giving

$$\alpha_{CPA} + \alpha_{\theta_T} + \alpha_R + \alpha_V = 1 \tag{3.5.14}$$

In this work, the numerical values in Table 3.5.1 are used. Initial choices were made based on values suggested in literature [80] [82], emphasising DCPA and TCPA. However, it was discovered that more weight had to be placed on the target bearing angle, absolute distance and approaching velocity to achieve the desired behaviour.

The configuration of the path following and static obstacle rewards listed in Table 3.4.1 have been reused here.

Parameter	Interpretation	Value
β_{CRI}	Scaling factor for overall risk level	10
β_{RL}	Scaling factor for the lower bound on distance	8
β_{RU}	Scaling factor for the upper bound on distance	18
θ_{PU}	Positive upper limit for the bearing angle θ_T	180°
θ_{PL}	Positive lower limit for the bearing angle θ_T	45°
θ_{NU}	Negative upper limit for the bearing angle θ_T	90°
θ_{NL}	Negative lower limit for the bearing angle θ_T	22.5°
α_{CPA}	Weighting of CPA membership function	0.3
α_{θ_T}	Weighting of target bearing angle membership function	0.2
α_R	Weighting of absolute distance membership function	0.3
α_V	Weighting of approaching velocity membership function	0.2
d_L	Minimal safe encounter distance	320 m
d_U	Absolute safe encounter distance	1500 m

Table 3.5.1: Reward configuration for the risk-based approach.

Chapter 4

Results

In this section, the results from the qualitative and risk-based implementations of the COLREGs are presented and evaluated.

4.1 Performance evaluation

In order to allow for comparison between the two approaches explored, a three-step evaluation process is employed. First, the agent's behaviour and performance in the training environment is assessed, and snippets from situations relevant to rules 14-16 of the COLREGs are presented. Next, two-vessel testing scenarios are constructed to specifically test for COLREG-compliance. Lastly, the agents are evaluated in AIS-based environments. These modes of assessment are described individually in the following subsections.

4.1.1 Performance in training environment

A natural starting point for performance evaluation is assessment of the agent's behaviour in its training environment. By collecting statistics on the collision rate, level of path completion, and reward, overall performance can be evaluated. These statistics both serve as a guide for when to stop the training, and as a point of comparison between approaches. Moreover, snippets are chosen from the simulations in the training environment in order to highlight the behaviour in situations where the COLREGs apply. The COLREGs cannot be said to always apply since the training environment often presents the agent with difficult and

unrealistic situations containing various static and dynamic obstacles which cannot be accurately subjected to the COLREGs.

4.1.2 Testing of COLREG-compliance

The next step in the testing process is subjecting the agent to scenarios specifically designed to capture COLREG-compliance. This is especially useful since it is difficult to find examples of scenarios perfectly showcasing COLREG-compliance in the training environment itself. Through simpler two-vessel scenarios, however, the agent's success can easily be quantified. One scenario to be tested is self-evident, namely the head-on scenario. In addition, two different crossing situations, one from starboard and one from the port side were chosen. For each scenario, the initial angles and path angles of the TS are varied slightly within a range of $\pm 5°$ of the default angles given in Table 4.1.1, which allows for repetitive testing in the respective scenarios.

Scenario	Default initial TS bearing angle	Default TS path angle
Head-on	0°	0°
Crossing from starboard	-30°	90°
Crossing from port	50°	-70°

Table 4.1.1: Default initial TS positions and path angles for COLREG-compliance tests.

It should be noted that the target ships have been modelled large in the testing scenarios to reflect the size of the large ships encountered in the AIS-based scenarios, and for visual clarity.

4.1.3 AIS-based testing

Lastly, three environments based on real-world high-fidelity terrain data are used to assess the overall performance of the agent. These environments were developed in [72] using AIS tracking data from the Trondheim Fjord area, and are distinctly different in nature. In the illustrations presented in the following, a dashed black line represents the desired vessel trajectory. Each TS is drawn at its initial position, and their trajectories are drawn as dotted red lines. Note that these are examples of spawned environments, and that the a set amount of target ships are chosen from the AIS database each time an instance of the specific scenario is created, giving different scenarios each time the environment is instantiated. Additionally,

the apparent density of TS trajectories does not directly reflect the amount of encounters, as this depends on the speed of each individual vessel.

The first AIS-based scenario is the **Trondheim** scenario exemplified in Fig. 4.1.1, in which the agent is required to cross a fjord of width ∼12 km while following a straight path. Doing so, it mainly meets crossing traffic consisting of larger vessels.

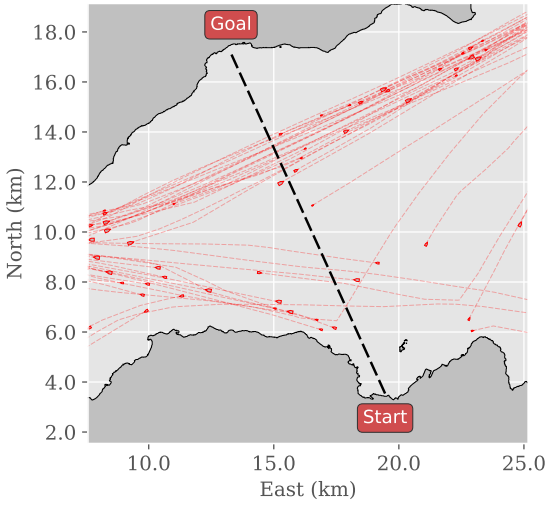


Figure 4.1.1: Map of the Trondheim test scenario.

In the challenging **Ørland-Agdenes** scenario in Fig. 4.1.2, the agent meets two-way traffic in a narrow fjord entrance region. To successfully complete the path, it must blend into the heavy traffic while avoiding head-on collisions. In addition, the ability to overtake other vessels is assessed. As in the Trondheim scenario, the vessels are mainly larger than the OS.

Lastly, the **Froan** scenario offers a demanding terrain with hundreds of small islands. As a result, it tests the ability of the agent to generalise to a challenging environment with a high density of static obstacles varying in size and shape. The area is less trafficked, and the vessels encountered are of similar size and speed to those of the OS. The scenario is presented in Fig. 4.1.3.

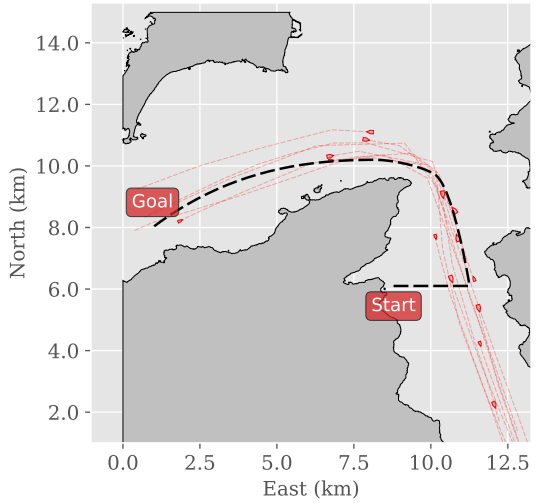


Figure 4.1.2: Map of the Ørland-Agdenes test scenario.

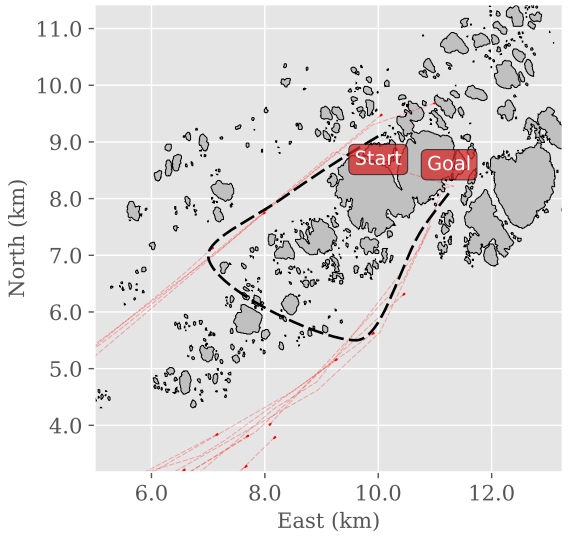


Figure 4.1.3: Map of the Froan test scenario.

4.2 Qualitative approach

Here, the results obtained using the qualitative approach are presented, following the procedure outlined.

4.2.1 Training

To choose a suited agent for testing, statistics from the training process were monitored. When the agent did no longer record collisions for over 500 episodes and the episode rewards stagnated after about 4000 episodes the training was halted. In Fig. 4.2.1 and 4.2.2 the frequency of collisions and progress are presented, showcasing a low collision rate and high path completion rate at the end of training. It is interesting to note that the collision rate quickly decreases, and after only 2000 episodes, the agent is already performing very well. The episode rewards are plotted in Fig. 4.2.3, showing a sharp increase in reward over the first 500 episodes, before slowly climbing and converging at a steady level.

Next, common collision avoidance maneuvers from the training environment are shown in Fig. 4.2.4. The snippets presented are representative of the agent's behavior in realistic encounter situations, which indicate that it is COLREG-compliant in the situations where the rules can be accurately discerned. As can be seen from Fig. 4.2.4a, the agent prefers to cross ahead of the TS at a distance of approximately 600 m to keep it on its port side in the head-on situation. Although not clearly defined in the COLREGs, there is a distance at which crossing ahead of other ships is considered safe, which has been indirectly communicated through the reward function. In situations such as this one, there is a trade-off between the penalty for keeping a TS on its starboard side and crossing ahead of it. Situations that distinctly require astern crossings, on the other hand, are handled with ease, as shown in Fig. 4.2.4b.

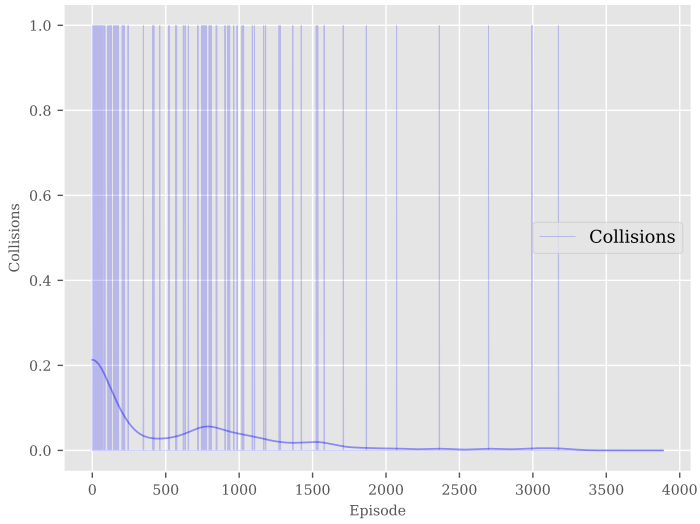


Figure 4.2.1: Collisions during training of qualitative agent. Each vertical line indicates a collision, and the line corresponds to the ratio of number of collisions to number of episodes.

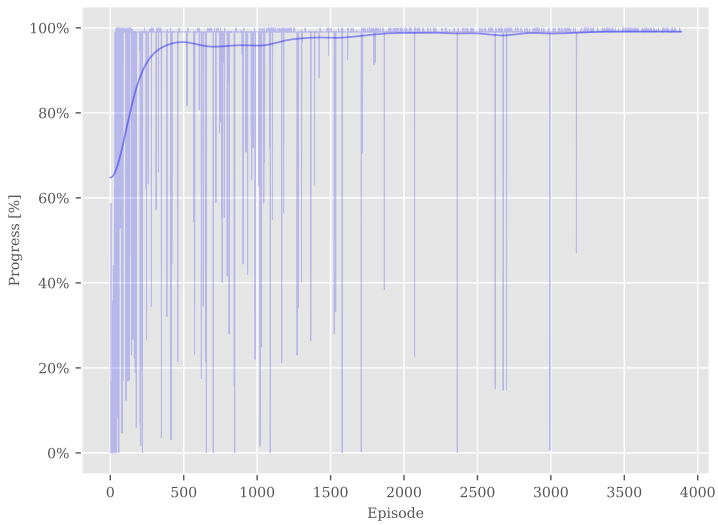


Figure 4.2.2: Path progress during training of qualitative agent.

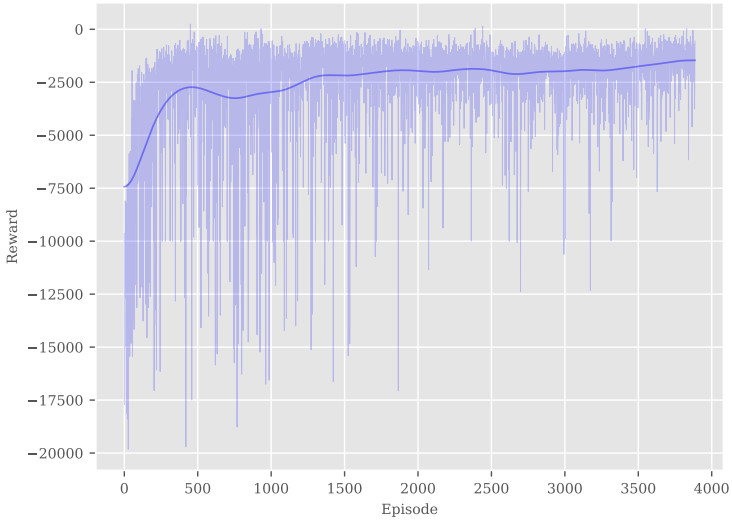
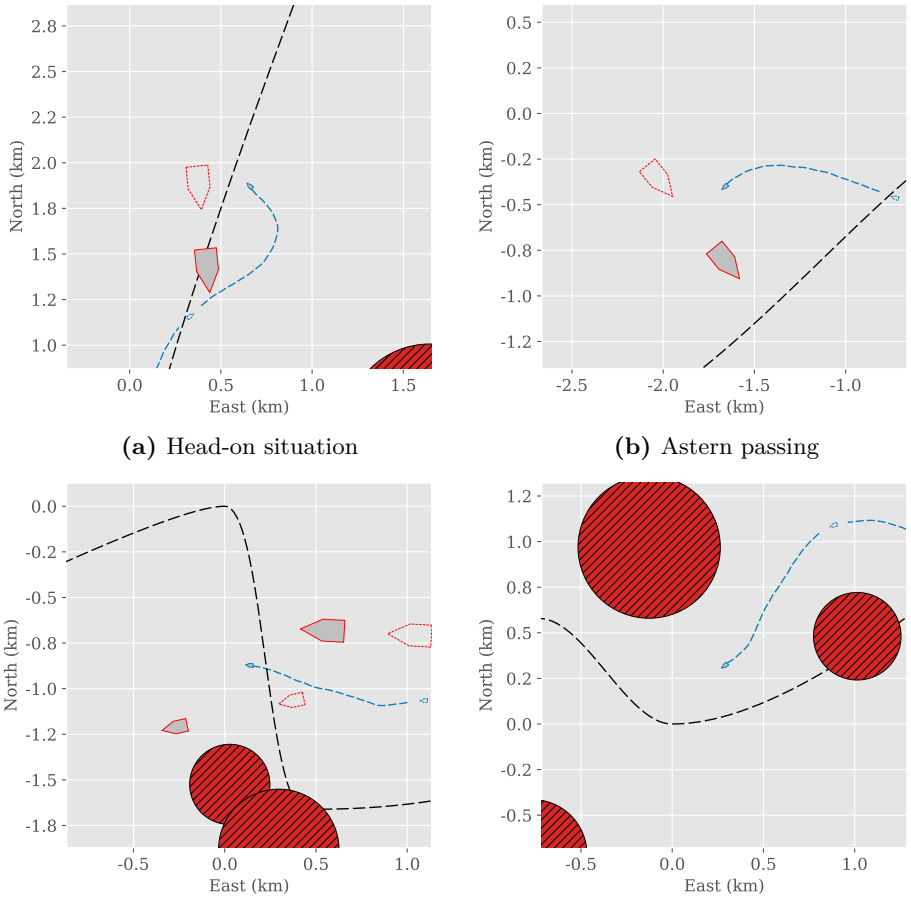


Figure 4.2.3: Accumulative reward during training of qualitative agent.



(a) Head-on situation

(b) Astern passing

(c) Overtaking

(d) Static COLAV

Figure 4.2.4: Qualitative agent performing common naval collision avoidance maneuvers in the training environment. The agent's trajectory is drawn as a blue dashed line, and the target ships with trajectories are drawn in red. The dotted vessel outlines show their positions 100 time steps prior to the present time.

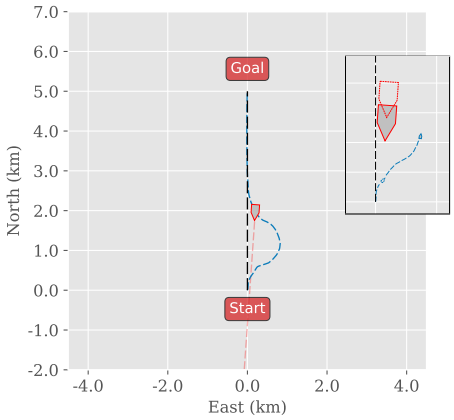
4.2.2 Testing of COLREG-compliance

Next, the agent was applied to the scenarios constructed for testing of COLREG-compliance, and the results can be seen in Fig. 4.2.5. Clearly, the agent adheres to the targeted COLREGs rules, and excellently follows the path once the TS is out of the sensor range. In Fig.4.2.5a, the OS adheres to Rule 14 by altering her course to starboard in a head-on situation. Further, as seen in Figs. 4.2.5b and 4.2.5c, the agent avoids crossing ahead of a TS when it can make a reasonable maneuver to cross astern, as described by Rules 15, 16, and 18. It was noted, however, that there is a "cut-off" when the TS approaches from an angle $\theta_T > 40°$. In these situations, it chooses to cross ahead, although with a good margin. This makes intuitive sense, as it effectively resolves the conflict without making sharp maneuvers. At the same time, it is important to note the ambiguity of the COLREGs in these cases, stating that the give-way vessel should "keep well clear".

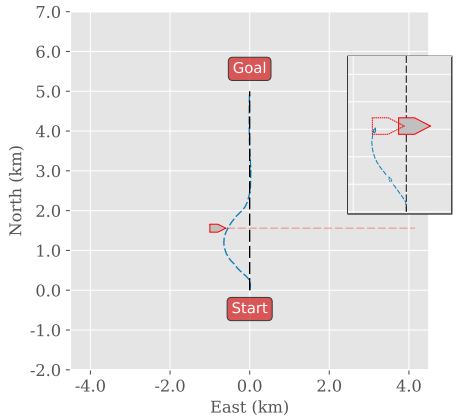
To thoroughly test COLREG-compliance, the three testing scenarios were simulated for 100 episodes each. For the head-on scenario, the incoming angle of the TS and the path angle α_p of the TS were varied such that $\theta_T \in [-5°, 5°]$ and $\alpha_p \in [-5°, 5°]$. Similarly, the incoming angle and path angle of the TS were varied with $\pm$ 5° relative to the angles utilised for the crossing from starboard and crossing from port scenarios in Fig. 4.2.5.

Scenario	Success rate
Head-on	100%
Crossing from starboard	100%
Crossing from port	100%

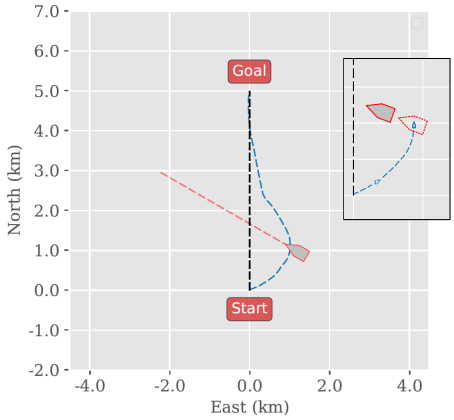
Table 4.2.1: Qualitative agent: results from repetitive testing of COLREGs with slightly varying scenarios, 100 episodes.



(a) Test scenario 1: Head on.



(b) Test scenario 2: Crossing from starboard.



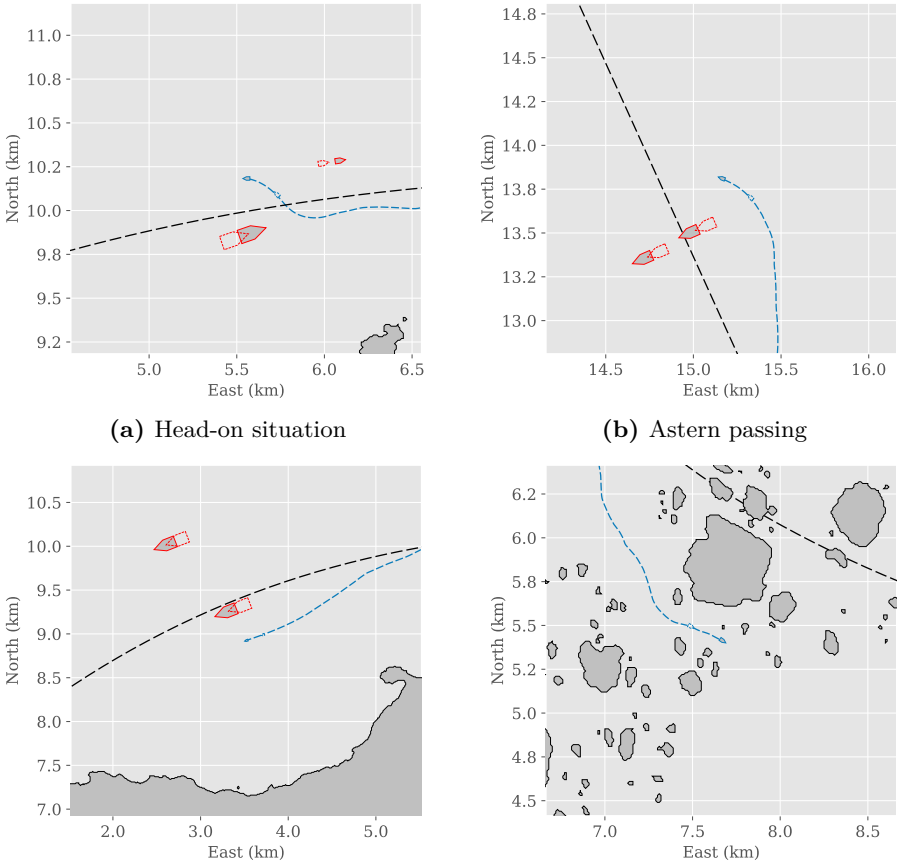
(c) Test scenario 2: Crossing from port.

Figure 4.2.5: Qualitative agent in COLREG-compliance test scenarios. The agent's trajectory is drawn as a blue dashed line, and the target ships with trajectories are drawn in red. The dotted vessel outlines show their positions 100 time steps prior to the present time.

4.2.3 Testing in AIS-based environment

Extending the testing to scenarios based on real-world AIS data, it can be seen that the agent behaves in a COLREG-compliant manner in situations where the COLREGs clearly define an expected behaviour. Some examples of this are presented in

Fig. 4.2.6, where situations similar to those chosen from the training environment (Fig. 4.2.4) were chosen for comparison. The main difference between the training environment and the AIS-based environment is the shapes and sizes of the static obstacles, which represent land and islands in the AIS-based environment. As seen in Fig. 4.2.6d, the agent has generalised sufficiently to tackle these scenarios with ease. Furthermore, overall trajectories undertaken by the agent in the Trondheim, Ørland-Agdenes and Froan scenarios can be seen in Fig. 4.2.7. Although the agent had no issue traversing the difficult terrain of Froan, it struggled when encountering target ships in restricted waters. The likely explanation for this is that the training environment does not reflect such situations properly for the agent to be prepared for them. For instance, in the training environment, the OS can always sail around a circular obstacle when encountering a TS close to such an obstacle. In the Froan scenario, this is not the case, and the OS easily gets lost attempting to find other ways to the goal. It should therefore be noted that in the scenario presented, the OS did not encounter another vessel after entering the narrow end section of the desired path, but is included to showcase the agent's ability to navigate in restricted waters in the absence of target ships.



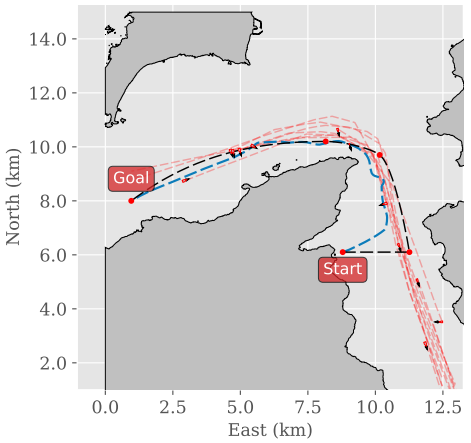
(a) Head-on situation

(b) Astern passing

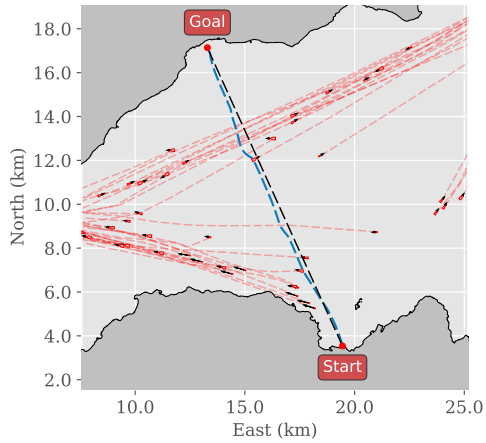
(c) Overtaking

(d) Static COLAV

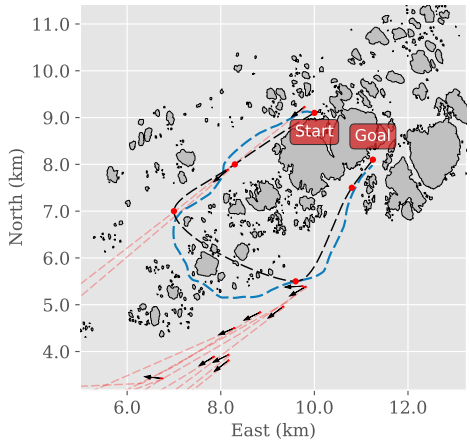
Figure 4.2.6: Qualitative agent performing common naval collision avoidance maneuvers in the AIS-based environment. The agent trajectory is drawn as a blue dashed line, and the target ships are drawn in red. The dotted vessel outlines show their positions 100 time steps prior to the present time.



(a) Qualitative agent's trajectory in the Ørland-Agdenes test scenario.



(b) Qualitative agent's trajectory in the Trondheim test scenario.



(c) Qualitative agent's trajectory in the Froan test scenario.

Figure 4.2.7: Qualitative agent trajectories in the AIS-based environments drawn as blue dashed lines. Target ships and trajectories are drawn in red.

4.3 Risk-based approach

In this section, the results obtained using the risk-based agent are showcased in same procedural manner as in Section 4.2, presenting training statistics, snippets from the training environment, COLREG-compliance tests, and testing in AIS-based environments.

4.3.1 Training

Although the design of the risk-based agent proved easier to tune due to the modularity of the fuzzy logic, the training process was less efficient from a collision point of view, and the convergence rate was significantly poorer than for the qualitative approach. This is seen in Fig. 4.3.1, where the pattern is denser than when training the qualitative agent. After about 4000 episodes, the collision rate dropped to near zero, however, and the progress rate rose to 100%, as shown in Fig. 4.3.2, at which point the training was stopped. The last training process statistic to evaluate is the accumulative reward, seen in Fig. 4.3.3. As was the case for the qualitative agent, the learning is steep initially, before the reward smoothens out and rises slowly. The agent used for further testing was chosen slightly before the training end point due to the slight dip in performance at the very end.

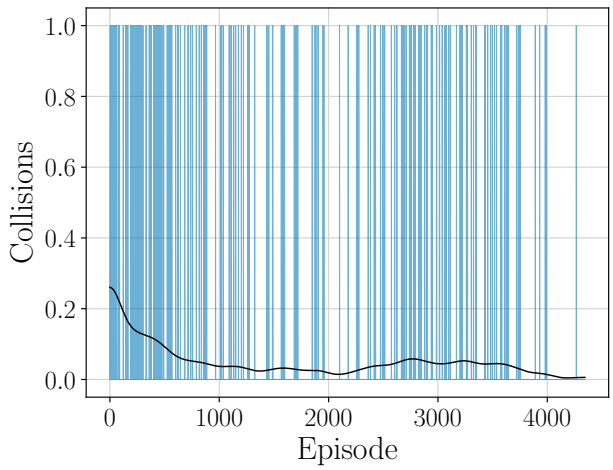


Figure 4.3.1: Collisions during training of risk-based agent. Each vertical line indicates a collision, and the line corresponds to the ratio of number of collisions to number of episodes.

The training statistics are of course of utmost importance to assess the overall per-

formance, but they do not reveal the level of COLREG-compliance, which must be evaluated separately. Snippets from the training environment have been included in Fig. 4.3.4, showcasing scenarios similar to those presented for the previous approach. It is clear that the agent behaves in a COLREG-compliant manner in situations precisely defined by the COLREGs by passing on the right in head-on situations, slowing down and passing astern instead of ahead, and allowing space between the OS and the TS during overtaking.

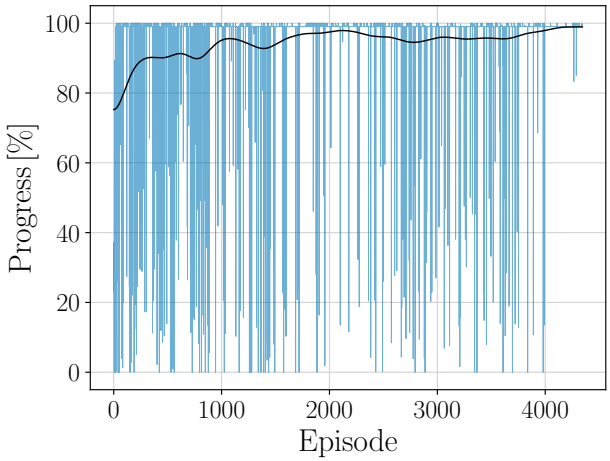


Figure 4.3.2: Path progress during training of risk-based agent.

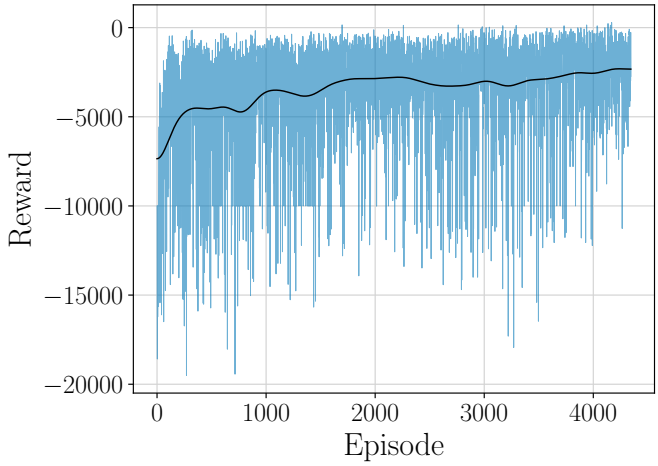
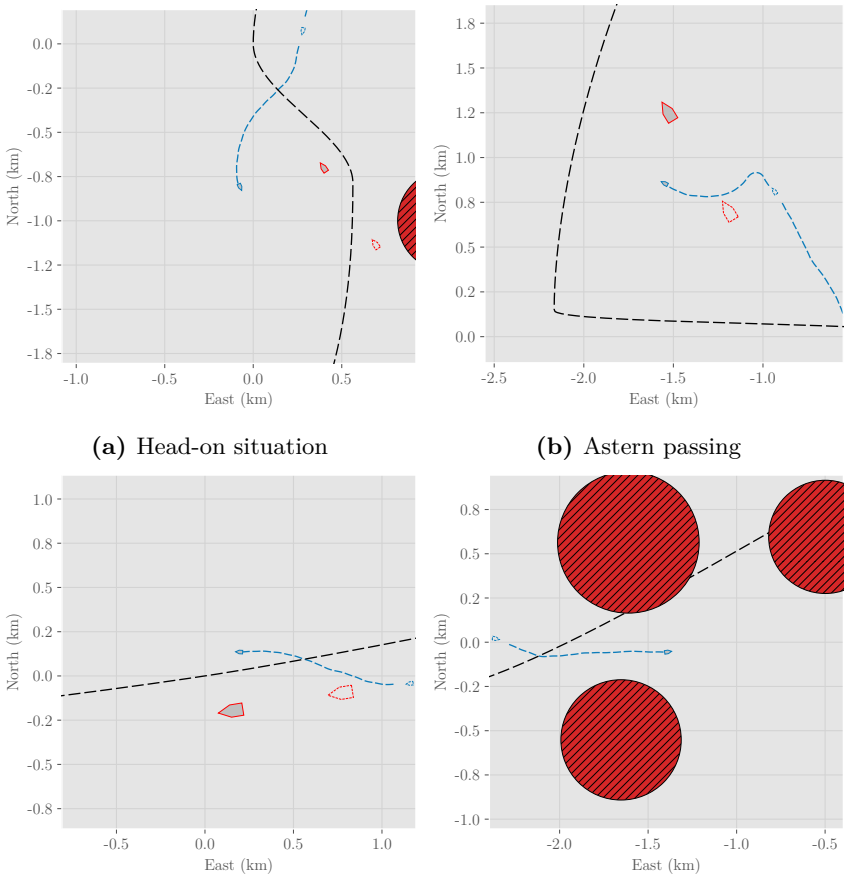


Figure 4.3.3: Accumulative reward during training of risk-based agent.



(a) Head-on situation

(b) Astern passing

(c) Overtaking

(d) Static COLAV

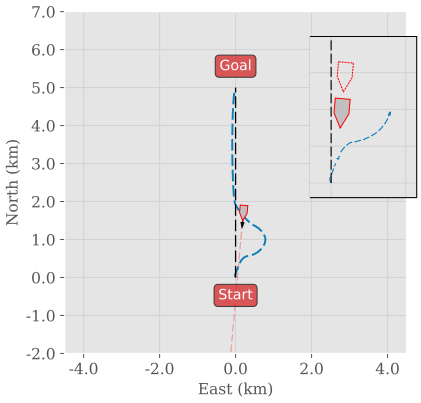
Figure 4.3.4: Risk-based agent performing common naval collision avoidance maneuvers in the training environment. The agent's trajectory is drawn as a blue dashed line, and the target ships with trajectories are drawn in red. The dotted vessel outlines show their positions 100 time steps prior to the present time.

4.3.2 Testing of COLREG-compliance

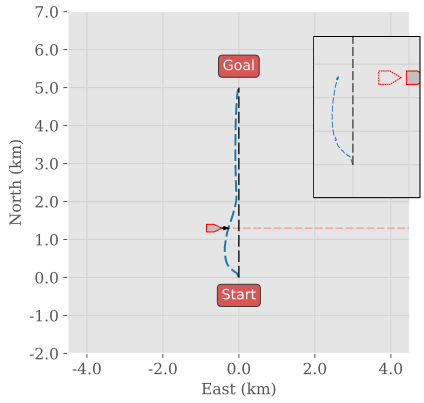
The next step in the evaluation process is COLREG-compliance testing with repetitive testing of each scenario. As seen in Fig. 4.3.5 the risk-based agent adheres to the COLREG rules 16-18; the OS keeps the TS on her port side in the head-on situation, and passes astern in both crossing situations. In addition, the agent follows the path well once the TS has been surpassed. Repetitive testing shows that these results are stable, as the correct behaviour was seen in 100% of the episodes for each testing scenario. These results are summarised in Table 4.3.1.

Scenario	Success rate
Head-on	100%
Crossing from starboard	100%
Crossing from port	100%

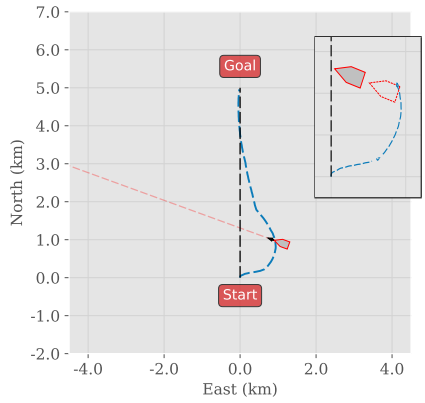
Table 4.3.1: Risk-based agent: results from repetitive testing of COLREGs with slightly varying scenarios, 100 episodes.



(a) Test scenario 1: Head on.



(b) Test scenario 2: Crossing from starboard.



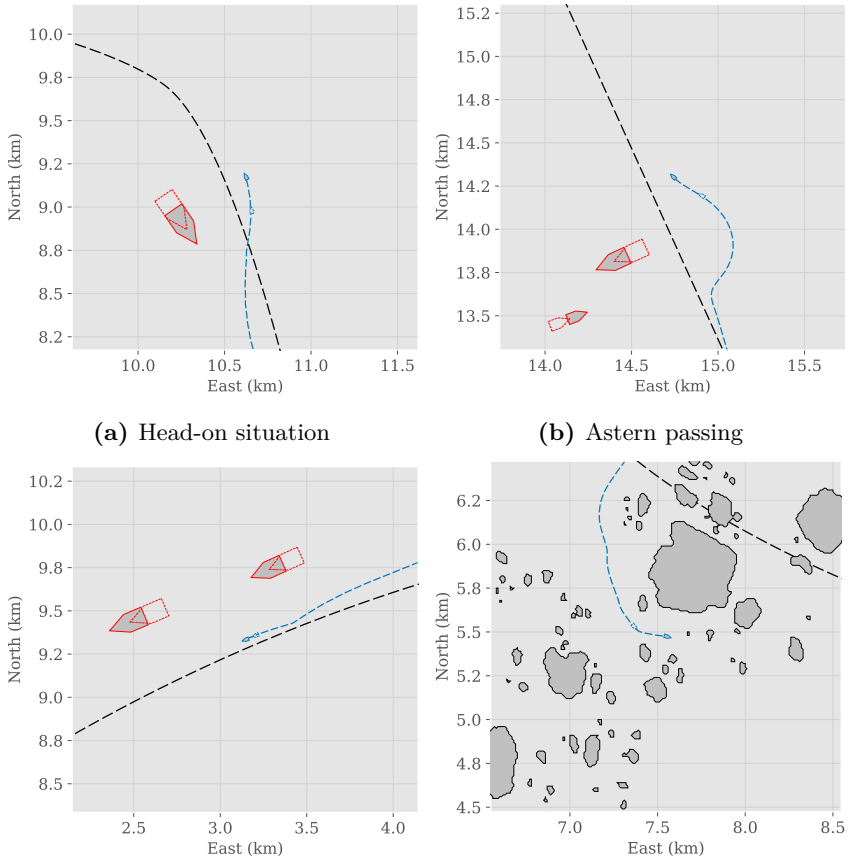
(c) Test scenario 2: Crossing from port.

Figure 4.3.5: Risk-based agent in COLREG-compliance test scenarios. The agent's trajectory is drawn as a blue dashed line, and the target ships with trajectories are drawn in red. The dotted vessel outlines show their positions 100 time steps prior to the present time.

4.3.3 Testing in AIS-based environment

Finally, the risk-based agent is assessed in the AIS-based environment. The snippets presented in Fig. 4.3.6 display COLREG-compliant behaviour and excellent static obstacle avoidance. As with the qualitative approach, we see in Fig. 4.3.6b that astern crossings are especially convincing. This is likely due to the lack of am-

biguity; the agent is keeping the TS on its port side while avoiding crossing ahead. Lastly, the overall trajectories are presented in Fig. 4.3.7, illustrating the agent's ability to dynamically follow a predetermined path in the face of static and moving obstacles. Moreover, the overall trajectories taken in the AIS-based environments are presented in Fig. 4.3.7, showcasing a great ability for dynamic path following and collision avoidance.



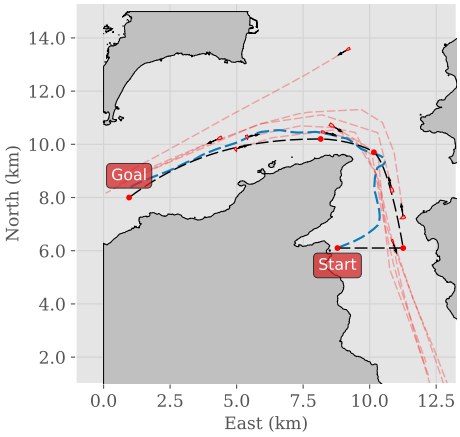
(a) Head-on situation

(b) Astern passing

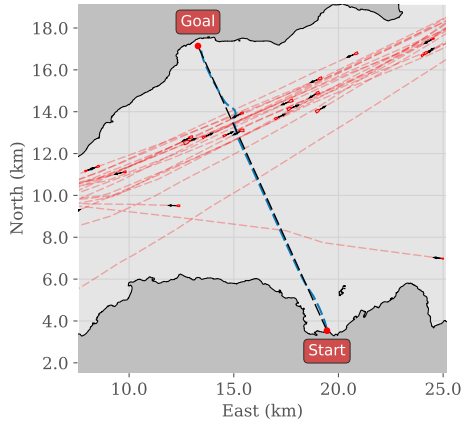
(c) Overtaking

(d) Static COLAV

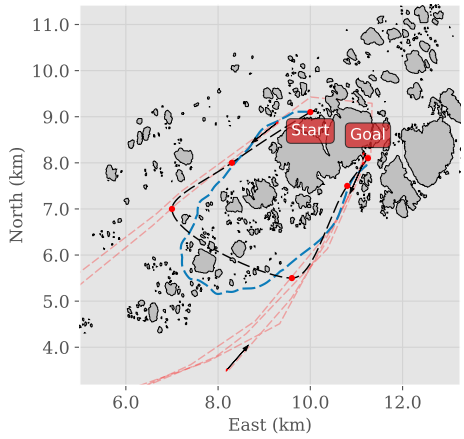
Figure 4.3.6: Risk-based agent performing common naval collision avoidance maneuvers in the AIS-based environment. The agent trajectory is drawn as a blue dashed line, and the target ships are drawn in red. The dotted vessel outlines show their positions 100 time steps prior to the present time.



(a) Risk-based agent's trajectory in the Ørland-Agdenes test scenario.



(b) Risk-based agent's trajectory in the Trondheim test scenario.



(c) Risk-based agent's trajectory in the Froan test scenario.

Figure 4.3.7: Risk-based agent trajectories in the AIS-based environments drawn as blue dashed lines. Target ships and trajectories are drawn in red.

Chapter 5

Conclusion and future work

Ending the thesis, this section contains a brief discussion and concluding remarks in Section 5.1, as well as suggested future work in Section 5.2.

5.1 Discussion and conclusion

The primary objective of the thesis was to investigate COLREG-compliance in a path following and collision avoidance system based on deep reinforcement learning through the comparison of a qualitative and a risk-based approach. As demonstrated in Section 4, both approaches produced COLREG-compliant behaviour when tested in scenarios relevant to COLREG rules 16-18, which are the main rules directly dictating desired collision avoidance behaviour. As such, DRL has proven to be up to the task of COLREG-compliant collision avoidance at sea.

Although the two approaches investigated produced similar end results, distinctions can be made regarding the ease of implementation and convergence, as well as general experience throughout the process. The qualitative approach, which was intended to be more intuitive and straight-forward to implement, turned out to be the hardest to tune. This can likely be attributed to the combination of different functions employed in the overall reward, including exponential functions, which made it challenging to predict how the tuning of the parameters would affect the agent's behaviour. On the other hand, the use of steeper functions in all likelihood contributed to the better convergence seen in the qualitative approach.

The opposite characteristics were observed for the risk-based approach, where the

tuning required less effort. A major reason for this is that the method is more transparent and explicit, consisting of relatively simple building blocks later combined through fuzzy comprehension. Moreover, being based on state-of-the-art collision risk measures, it was easier to be convinced by the approach. These virtues of transparency and trust cannot be downplayed for safety-critical applications such as collision avoidance. Unfortunately, the unintended effect was a slower convergence rate.

How, then, can the first research question be answered: which of the approaches implemented shows the greatest potential for COLREG-compliant collision avoidance? The high convergence rate of the qualitative approach provides evidence that it captures the COLREGs well, and can be said to be the preferred method of the two at face value. However, as transparency and trust are vital in any collision avoidance system, the risk-based method seems to hold the greatest potential in the long run. This is especially true as it is reasonable to think that the convergence issues of the risk-based method can be resolved by introducing a mapping from the collision index to the reward. Doing so, the features of the risk level can be accentuated for faster learning.

In either case, the COLREGs must be adapted for machine readability and interpretability, clearly defining the required behaviour in different environments and situations. Until then, it will be impossible to accurately assess the success of any autonomous vessel and claim that it is fully COLREG-compliant. Having said that, the results obtained in this work bear witness to the ability of DRL to handle COLREG rules. Resultantly, once the COLREGs are modernised for digital applications, DRL can be expected to produce COLREG-compliant and autonomous collision avoidance systems. In other words, DRL is suited for COLREG-compliant navigation at sea – answering the second research question.

5.2 Future work

5.2.1 Multi-agent environments

In the work and results presented in this thesis, the agent is interacting with vessels that are blindly following predetermined paths. Needless to say, such a setup is not very realistic, as most encountered target vessels would make attempts to avoid collisions. Therefore, extending the work to a multi-agent environment would be a natural next step in the process of investigating COLREG-compliance in a DRL framework.

The simulation environment provided as a starting point was expanded to begin exploring training and testing in multi-agent scenarios. However, since the focus here has been on COLREG-compliance, and the topic of multi-agent DRL systems is worthy of an entire master thesis of its own, it has been left for future students to tackle.

Some of the challenges posed in multi-agent systems, especially in combination with the current framework, are: 1) the Sable Baselines package providing the DRL algorithms is not currently compatible with multi-agent systems, and 2) the high computational complexity of the current virtual sensor suite would lead to a significant slowing of simulation times. This is the case since the time step and/or maximum velocity must be decreased to ensure predictability while training multiple agents at once. As a result, it might be necessary to significantly alter the framework, finding inspiration in implementations of PPO for multi-agent systems and swarm robotics, e.g. TensorSwarm [83].

5.2.2 Explainability and stability analysis

A major point of criticism towards AI methods such as DRL is the lack of explainability and analysability. Neural networks are considered to be *black boxes*, meaning that their inner workings are difficult to access and evaluate. Furthermore, stability analyses are necessary to assess the robustness of any control system in a safety-critical application. Consequently, it is crucial to move towards less opaque AI.

As a response to the criticism, the field of *explainable AI* (XAI) has emerged, seeking to develop methods to increase the transparency of AI systems. XAI techniques can largely be separated into two categories based on the type of interpretability they bring to the table: *perceptive interpretability* or *interpretability by mathematical structures* [84]. A large group of techniques within perceptive interpretability relies on saliency to express the basis of decisions to the user, for instance by creating heat maps or presenting the probabilities of different outputs. Techniques based on mathematical structures attempt to understand and express the underlying mechanisms of the neural network. Examples of this are clustering methods and subspace analysis, aiming to determine the underpinnings of the decisions made by the neural network.

XAI is only one side of the coin, however, as it does not encompass the domain of stability analysis. As such, XAI alone does not bridge the gap between DRL and real-life, satefy-critical applications, which require robust control systems. Al-

though this is a relatively unexplored area, some methods have been proposed. In [85], quadratic constraints and semidefinite programming is used to perform a robustness analysis based on properties such as the boundedness and monotonicity of the network's activation function. Another possibility is extracting analysable equations from the neural network via symbolic regression, as proposed in [86] this year. Symbolic regression could also potentially be used to express an agent's learning in the form of ordinary differential equations.

5.2.3 Realistic environments

As outlined in Section 2.1, a fundamental assumption here has been calm sea – the absence of environmental disturbances such as wind, ocean currents, and waves. An interesting extension would be to include such disturbances to determine whether the DRL system is able to deal with them. Based on the findings and experiences working with DRL (and the PPO algorithm specifically), it seems likely that a DRL system would efficiently counteract moderate disturbances. In [78], good path following results are obtained for a DRL-based path following system in the face of disturbances, providing evidence for this. A way to further aid efficient learning is to include key information about the environment, such as the direction of current, in the observation vector. However, if the goal is to also include the environmental data in the risk assessment, there is a longer way to go. In their 2019 review of collision risk measures, Ozturk and Cicek note that little consideration of environmental factors is made in state-of-the-art methods for assessing collision risk [87]. Filling that gap in research is therefore needed before applying a purely risk-based DRL approach to the problem.

Bibliography

[1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016, accessed: 2019-11-02.

[2] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable Baselines," https://github.com/hill-a/stable-baselines, 2018.

[3] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[4] S. Gillies *et al.*, "Shapely: manipulation and analysis of geometric objects," toblerity.org, 2007–. [Online]. Available: https://github.com/Toblerity/Shapely

[5] E. Meyer, A. Heiberg, A. Rasheed, and O. San, "COLREG-Compliant Collision Avoidance for Unmanned Surface Vehicle using Deep Reinforcement Learning," *arXiv e-prints*, p. arXiv:2006.09540, Jun. 2020.

[6] DNV GL, "The ReVolt – A new inspirational ship concept," https://www.dnvgl.com/technology-innovation/revolt/index.html, 2020, accessed: 2020-27-05.

[7] KONGSBERG, "Autonomous ship project, key facts about Yara Birkeland," https://www.kongsberg.com/no/maritime/support/themes/autonomous-ship-project-key-facts-about-yara-birkeland/, 2020, accessed: 2020-27-05.

[8] Autoship, "AUTOSHIP – Autonomous Shipping Initiative for European Waters," https://www.autoship-project.eu/, 2020, accessed: 2020-06-10.

[9] T. A. Dingus, F. Guo, S. Lee, J. F. Antin, M. Perez, M. Buchanan-King, and J. Hankey, "Driver crash risk factors and prevalence evaluation using naturalistic driving data," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 113, no. 10, pp. 2636–2641, 2016.

[10] P. Thomas, A. Morris, R. Talbot, and H. Fagerlind, "Identifying the causes of road crashes in Europe," *Annals of Advances in Automotive Medicine*, vol. 57, no. November 2014, pp. 13–22, 2013.

[11] European Maritime Safety Agency (EMSA), "Annual Overview of Marine Casualties and Incidents 2018," http://www.emsa.europa.eu/news-a-press-centre/external-news/item/3406-annual-overview-of-marine-casualties-and-incidents-2018.html, 2018, accessed: 2019-11-05.

[12] A. Skredderberget, "The first ever zero emission, autonomous ship," https://www.yara.com/knowledge-grows/game-changer-for-the-environment/, 2018, accessed: 2019-11-12.

[13] B. Lin and C. H. Huang, "Comparison between arpa radar and AIS characteristics for vessel traffic services," *Journal of Marine Science and Technology*, vol. 14, no. 3, pp. 182–189, 2006.

[14] Q. Xu and N. Wang, "A survey on ship collision risk evaluation," *Promet - Traffic&Transportation*, vol. 26, no. 6, pp. 475–486, 2014.

[15] DNV GL, "Digital twins and sensor monitoring," https://www.dnvgl.com/expert-story/maritime-impact/Digital-twins-and-sensor-monitoring.html, 2019, accessed: 2020-02-25.

[16] A. Aniculaesei, D. Arnsberger, F. Howar, and A. Rausch, "Towards the verification of safety-critical autonomous systems in dynamic environments," *Electronic Proceedings in Theoretical Computer Science, EPTCS*, vol. 232, pp. 79–90, 2016.

[17] International Maritime Organization (IMO), "Convention on the International Regulations for Preventing Collisions at Sea, 1972 (COLREGs)," http://www.imo.org/en/About/Conventions/ListOfConventions/Pages/COLREG.aspx, accessed: 2019-09-16.

[18] C. K. Tam, R. Bucknall, and A. Greig, "Review of collision avoidance and path planning methods for ships in close range encounters," *Journal of Navigation*, vol. 62, no. 3, pp. 455–476, 2009.

[19] T. Statheros, G. Howells, and K. McDonald-Maier, "Autonomous ship collision avoidance navigation concepts, technologies and techniques," *Journal of Navigation*, vol. 61, no. 1, pp. 129–142, 2008.

[20] T. A. Johansen, T. Perez, and A. Cristofaro, "Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3407–3422, dec 2016.

[21] B.-O. H. Eriksen, "Collision Avoidance and Motion Control for Autonomous Surface Vehicles," Ph.D. dissertation, Norwegian University of Science and Technology, 2019.

[22] R. Soloperto, J. Kohler, F. Allguwer, and M. A. Muller, "Collision avoidance for uncertain nonlinear systems with moving obstacles using robust Model Predictive Control," *18th European Control Conference (ECC)*, pp. 811–817, 2019.

[23] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[24] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGS, using velocity obstacles," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 110–119, jan 2014.

[25] M. R. Benjamin, J. J. Leonard, J. A. Curcio, and P. M. Newman, "A Method for Protocol-Based Collision Avoidance between Autonomous Marine Surface Crafts," *Journal of Field Robotics*, vol. 29, no. 4, pp. 554–575, 2006.

[26] K. Woernner, "COLREGs-Compliant Autonomous Collision Avoidance Using Multi-Objective Optimization with Interval Programming," *Naval Engineers Journal*, vol. 126, no. 2, pp. 64–65, 2014.

[27] E. Serigstad, B. O. H. Eriksen, and M. Breivik, "Hybrid Collision Avoidance for Autonomous Surface Vehicles," *IFAC-PapersOnLine*, vol. 51, no. 29, pp. 1–7, 2018.

[28] D. Fox, B. Wolfram, and T. Sebastian, "The Dynamic Window Approach to Collision Avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, pp. 137–146, 1997.

[29] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic

Determination of Minimum Cost Paths," *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 4, pp. 514–532, 1968.

[30] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *IEEE*, pp. 500–505, 1985.

[31] L. Zhou and W. Li, "Adaptive Artificial Potential Field Approach for Obstacle Avoidance Path Planning," in *Proceedings - 2014 7th International Symposium on Computational Intelligence and Design, ISCID 2014*, vol. 2. Institute of Electrical and Electronics Engineers Inc., apr 2015, pp. 429–432.

[32] R. Smierzchalski, "Evolutionary trajectory planning of ships in navigation traffic areas," *Journal of Marine Science and Technology*, vol. 4, no. 1, pp. 1–6, 1999.

[33] D.-G. Kim, K. Hirayama, and T. Okimoto, "Ship Collision Avoidance by Distributed Tabu Search," *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, vol. 9, no. 1, pp. 23–29, apr 2015.

[34] Q. Xu, C. Zhang, and L. Zhang, "Deep convolutional neural network based unmanned surface vehicle maneuvering," *Proceedings, 2017 Chinese Automation Congress (CAC)*, no. 2, pp. 878–881, 2017.

[35] A. B. Martinsen and A. M. Lekkas, "Straight-Path Following for Underactuated Marine Vessels using Deep Reinforcement Learning," *IFAC*, 2018.

[36] ——, "Curved Path Following with Deep Reinforcement Learning: Results from Three Vessel Models," *OCEANS 2018 MTS/IEEE Charleston, OCEAN 2018*, 2019.

[37] I. J. Vallestad, "Path Following and Collision Avoidance for Marine Vessels with Deep Reinforcement Learning," Master's thesis, Norwegian University of Science and Technology, 2019.

[38] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *International Conference on Learning Representations 2016*, sep 2015. [Online]. Available: http://arxiv.org/abs/1509.02971

[39] L. Zhao and M. I. Roh, "COLREGs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Engineering*, 2019.

[40] E. Meyer, H. Robinson, A. Rasheed, and O. San, "Taming an autonomous surface vehicle for path following and collision avoidance using deep

reinforcement learning," *IEEE Access*, pp. 1–16, 2019. [Online]. Available: http://arxiv.org/abs/1912.08578

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *CoRR*, pp. 1–12, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[42] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer, 2008, vol. 1.

[43] International Maritime Organization (IMO), "AIS Transponders," http://www.imo.org/en/OurWork/Safety/Navigation/Pages/AIS.aspx, 2020, accessed: 2019-09-16.

[44] J. Ding, J. H. Gillula, H. Huang, M. P. Vitus, W. Zhang, , and C. J. Tomlin, "Hybrid Systems in Robotics: Toward Reachability-Based Controller Design," *IEEE Robotics & Automation Magazine*, 2011.

[45] Ø. A. G. Loe, "Collision Avoidance for Unmanned Surface Vehicles," Ph.D. dissertation, Norwegian University of Science and Technology, 2008. [Online]. Available: http://ntnu.diva-portal.org/smash/record.jsf?pid=diva2:347606

[46] G. Casalino, A. Turetta, and E. Simetti, "A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields," *OCEANS '09 IEEE Bremen: Balancing Technology with Future Needs*, pp. 1–8, 2009.

[47] P. Svec, B. C. Shah, I. R. Bertaska, J. Alvarez, A. J. Sinisterra, K. Von Ellenrieder, M. Dhanak, and S. K. Gupta, "Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3871–3878, 2013.

[48] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Packt Publishing, 2017, ISBN: 978-1-1199-9149-6.

[49] SNAME, The Society of Naval Architecture and Marine Engineers, "Nomenclature for treating the motion of a submerged body through a fluid"," 1950.

[50] G. Lie, R. Zejian, G. Pingshu, and C. Jing, "Advanced emergency braking controller design for pedestrian protection oriented automotive collision avoidance system," *The Scientific World Journal*, 2014.

[51] A. Lazarowska, "Research on algorithms for autonomous navigation of ships," *WMU Journal of Maritime Affairs*, pp. 341–358, 2019.

[52] "COLREGS - International Regulations for Preventing Collisions at Sea ," http://www.mar.ist.utl.pt/mventura/Projecto-Navios-I/IMO-Conventions%20(copies)/COLREG-1972.pdf, 1972, accessed: 2019-11-05.

[53] A. N. Cockcroft, "The Estimation of Collision Risk for Marine Traffic," *Journal of Navigation*, vol. 34, no. 1, p. 145–147, 1981.

[54] E. M. Goodwin, "Marine Encounter Rates," *Journal of Navigation*, vol. 31, no. 3, p. 357–369, 1978.

[55] R. Szlapczynski and J. Szlapczynska, "Review of ship safety domains: Models and applications," *Ocean Engineering*, vol. 145, pp. 277–289, 2017. [Online]. Available: https://doi.org/10.1016/j.oceaneng.2017.09.020

[56] A. Rawson, E. Rogers, D. Foster, and D. Phillips, "Practical application of domain analysis: Port of london case study," *Journal of Navigation*, vol. 67, no. 2, pp. 193–209, 2014.

[57] Y. Wang and H. C. Chin, "An Empirically-Calibrated Ship Domain as a Safety Criterion for Navigation in Confined Waters," *Journal of Navigation*, vol. 69, no. 2, pp. 257–276, 2016.

[58] P. V. Davis, M. J. Dove, and C. T. Stockel, "A Computer Simulation of Marine Traffic Using Domains and Arenas," *Journal of Navigation*, vol. 33, no. 2, p. 215–222, 1980.

[59] G. Almog, "Traditional Recruiting Isn't Enough: How AI Is Changing The Rules In The Human Capital Market," https://www.forbes.com/sites/groupthink/2018/02/09/traditional-recruiting-isnt-enough-how-ai-is-changing-the-rules-in-the-human-capital-market/#5e6d508274a7, 2018, accessed: 2019-10-31.

[60] A. Bhola, "Knowledge Transfer in Reinforcement Learning," https://regressionist.github.io/2019-05-13-Reinforcement-Learning/, 2019, accessed: 2019-11-15.

[61] H. Chinaei and B. Chaib-draa, *Building Dialogue POMDPs from Expert Dialogues: An end-to-end approach.* Springer, 2016, ISBN: 978-3-3192-6200-0.

[62] R. Bellman, "The Theory of Dynamic Programming," The RAND Corporation, pp. 503–515, 1954.

[63] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, A. Mohiuddin, R. Sepassi, G. Tucker, and H. Michalewski, "Model-Based Reinforcement Learning for Atari," *International Conference on Learning Representations 2020*, 2019. [Online]. Available: http://arxiv.org/abs/1903.00374

[64] C. Watkins, "Learning from delayed rewards," *Robotics and Autonomous Systems*, vol. 15, no. 4, pp. 233–235, 1989.

[65] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," *Cambridge University Engineering Department*, 1994.

[66] V. Heidrich-Meisner, M. Lauer, C. Igel, and M. Riedmiller, "Reinforcement learning in a nutshell," *ESANN 2007 Proceedings - 15th European Symposium on Artificial Neural Networks*, pp. 277–288, 2007.

[67] P. Dangeti, *Statistics for Machine Learning*. Wiley-Blackwell, 2011, ISBN: 978-1-78829-575-8.

[68] P. Lucidarme, "Simplest perceptron update rules demonstration," https://www.lucidarme.me/simplest-perceptron-update-rules-demonstration//, 2020, accessed: 2019-11-15.

[69] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust Region Policy Optimization John," *Proceedings of the 31 st International Conference on Machine Learning*, vol. 37, 2015.

[70] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *31st International Conference on Machine Learning, ICML 2014*, vol. 1, pp. 605–619, 2014.

[71] E. Meyer, H. Robinson, A. Rasheed, and O. San, "Taming an Autonomous Surface Vehicle for Path Following and Collision Avoidance Using Deep Reinforcement Learning," *IEEE Access*, vol. 8, pp. 41 466–41 481, 2020.

[72] E. Meyer, "On Course Towards Model-Free Guidance: A Self-Learning Approach To Dynamic Collision Avoidance for Autonomous Surface Vehicles," Master's thesis, Norwegian University of Science and Technology, 2020.

[73] R. Skjetne, Ø. Smogeli, and T. I. Fossen, "Modeling, Identification and Adaptive Maneuvering of Cybership II: A complete design with experiments," *CAMS*, pp. 203–208, 2004.

[74] R. Skjetne, "The Maneuvering Problem," Ph.D. dissertation, Norwegian University of Science and Technology, 2005.

[75] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018.

[76] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation Matters in Deep Policy Gradients: A case study on PPO and TRPO," *ICLR 2019*, 2019.

[77] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[78] A. B. Martinsen, "End-to-end training for path following and control of marine vehicles," Master's thesis, Norwegian University of Science and Technology, 2018. [Online]. Available: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2559484/18542{_}FULLTEXT.pdf?sequence=1{&}isAllowed=y

[79] L. Gang, Y. Wang, Y. Sun, L. Zhou, and M. Zhang, "Estimation of vessel collision risk index based on support vector machine," *Advances in Mechanical Engineering*, vol. 8, no. 11, pp. 1–10, 2016.

[80] S. Chen, R. Ahmad, B. G. Lee, and D. H. Kim, "Composition ship collision risk based on fuzzy theory," *Journal of Central South University*, vol. 21, no. 11, pp. 4296–4302, 2014.

[81] G.-K. Park, J. L. R. M. Benedictos, S. C. Shin, and I. Nam-Kyun, "Design of a Fuzzy-CBR Support System for Ship ' s Collision Avoidance," *SCIS&ISIS2006*, pp. 240–248, 2006.

[82] Q. Yan, "A Model for Estimating the Risk Degrees of Collisions," *Journal of Wuhan University of Technology (Transportation Science & Engineering)*, vol. 26, no. 2, 2002.

[83] A. Pasternak, "TensorSwarm," https://github.com/TensorSwarm/TensorSwarm, 2017–.

[84] E. Tjoa and C. Guan, "A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI," *arXiv e-prints*, p. arXiv:1907.07374, Jul. 2019.

[85] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming," *arXiv e-prints*, p. arXiv:1903.01287, Mar. 2019.

[86] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, "Discovering Symbolic Models from Deep Learning with Inductive Biases," *arXiv e-prints*, p. arXiv:2006.11287, Jun. 2020.

[87] U. Ozturk and K. Cicek, "Individual collision risk assessment in ship navigation: A systematic literature review," *Ocean Engineering*, vol. 180, pp. 130–143, 2019.

Appendix A

Feasibility pooling algorithm

Algorithm 1 Feasibility pooling for rangefinder sensors [71].

Require:
Vessel width $W \in \mathbb{R}^+$
Angle between neighboring sensors θ
Sensor rangefinder measurements for current sector $\boldsymbol{x} = \{x_1, \ldots, x_n\}$
procedure FEASIBILITYPOOLING($\boldsymbol{x}$)
Initialize $\mathcal{I}$ to be the indices of $\boldsymbol{x}$ sorted in ascending order according to the measurements x_i
for $i \in \mathcal{I}$ **do**
Arc-length $d_i \leftarrow \theta x_i$
Opening-width $y \leftarrow d_i/2$
Opening was found $s_i \leftarrow false$
for $j \leftarrow 0$ to n **do**
if $x_j > x_i$ **then**
$y \leftarrow y + d_i$
if $y > W$ **then**
$s_i \leftarrow true$
break
else
$y \leftarrow y + d_i/2$
if $y > W$ **then**
$s_i \leftarrow true$
break
$y \leftarrow 0$
if s_i is $false$ **then return** x_i

Amalie Heiberg

COLREG-Compliance for ASVs using Deep Reinforcement Learning

NTNU
Norwegian University of
Science and Technology